Table of Contents

Job Traceability in CMS

Introduction

An important security auditing concept is *traceability* -- the ability to determine who was using a specific computing resource during a given time period. For some sites, traceability is a local legal mandate; for others, it's just a part of meeting their WLCG MoU commitments. Ultimately, **traceability is a site responsibility**; this responsibility can be delegated back to the VO.

In this page, we discuss a bit of the historical context about why this delegation is done and then discuss CMS-specific methods to trace our use of payload jobs.

Historical Context

Traceability at sites was controlled at the Unix user level. Users authenticated with the site at the batch submission point as a unique Unix user. By examining the batch system logs, the timestamps of files created, and the ownership of running processes, and knowing how to reverse map the Unix user to an individual, the site could map resource usage back to individuals.

The WLCG VOs, however, rely on the multi-user pilot job model; the batch system only sees the pilot. The pilot subsequently uses resources allocated by the batch system and delegates them to multiple users. There is no longer a unique mapping from batch system activity under a single user account back to an individual.

The first step toward traceability was glexec. This tool allowed the site to setup mappings of user grid identities to Unix users and provided a mechanism for the pilot to launch processes as the Unix user. Thus, we reused the traditional Unix security model to trace activity to a grid identity. **Note**, three important aspects of this model:

- Sites *must* setup glexec such that grid identities mapped to a unique Unix user per individual, introducing state and possibility of misconfiguration into the system. Misconfigured sites have significantly reduced traceability capabilities.
- Sites rely on the VO to utilize glexec correctly and to match grid identities to the individuals' executable payload.

Hence, even with glexec, sites delegate traceability responsibility to the VO. In the case of glexec, the relevant activity logs are kept *at the site* (as they are produced by the glexec executable).

Traceability Today

For various reasons (some technical, some organizational), the glexec based approach has fallen out of favor. Instead, the sites tend to rely on the record keeping of the VO: given a host and time -- or a site batch system ID -- the VO should be able to provide a list of individuals who were using the computing resources. The sole user of the glexec model is CMS; CMS is beginning to switch to singularity to meet its isolation capabilities and retire glexec support.

A few outcomes of this transition:

- VOs may track the VO identity of the user, not the grid or global identity. There may be additional steps required to correlate usage across VOs - or map from the VO identity to the individual.
- VOs become responsible for retaining the relevant logs about user activity within pilots.
- It is more obvious to sites they have delegated the responsibility of traceability to the VO. This was true with glexec, but significantly more visible due to the fact the logs are owned by the VO.

Note that there are several subjective items in our definition of **traceability**:

- What defines the granularity of 'computing resource' we need to trace - is it the site? The physical piece of hardware? The container the pilot was launched within?
    - In the case where pilots are launched inside containers, tracking between pilots is problematic.
- What is the time period / granularity? Do we need millisecond accuracy? Day-level?

For this document, we aim to provide the following:

- Given a specific GMT time and a pilot ID or UTS (host) name,
- Provide a list of all CMS users whose code we executed within the corresponding UTS name at that site,
- Starting at the specified time plus 48 hours.
- This capability should be possible for any CMS activity within the prior 90 days.

[Specifically, by using the UTS name as the identifier, we do not try to achieve the ability to link between containers on the same pilot.]

Traceability with CMS's ElasticSearch

Login to https://es-cms.cern.ch; this should be accessible to anyone within CMS.

The "discovery" tab will provide a list of individual payload jobs. First, adjust the time picker (upper right-hand-side) to the time period in question. It is suggested to broaden the time window's begin and end timestamp by 8 hours. **Note the time is based on the browser timezone** not GMT; for example, central standard time is 5 hours behind GMT. If the request comes in based on GMT, adjust accordingly.

Next, filter the corresponding jobs.

- Use the `Site` column to specify the site. This is the CMS site name (`T2_CH_CERN`, not WLCG's `CERN-PROD`).
- Simply type the hostname if given a hostname to search for.
- If given a site job ID, using the `GLIDEIN_SiteWMS_JobId` column. Additionally, `GLIDEIN_SiteWMS_Queue` would specify the queue or CE the job was submitted to (at some sites, `JobId` is insufficient for uniqueness).
- Multiple filters may be concatenated with `AND` (case-sensitive).

Example filters:

- `b63a6819bf.cern.ch AND Site:T2_CH_CERN`
- `GLIDEIN_SiteWMS_JobId:5581127.0 AND Site:T2_CH_CERN`

This will result in a list of jobs that **finished** on that resource within the given timeframe. ElasticSearch indexes payload jobs on completion time, which is why one wants to broaden the query time window as we will also want to take into consideration job start time.

A few relevant columns to consider:

- `x509userproxysubject`, `x509UserProxyEmail`, or `CRAB_UserHN`: the identity of the payload job's owner. Depending on job type and the certificate contents, not all may be populated. We expect `x509userproxysubject` is always present.
- `JobCurrentStartDate`: When the payload job began to execute.
- `CompletionDate`: When the payload job was completed and exited the queue.

Ideally, we want to report not only the potentially affected users running within that pilot -- but also user whose proxy may have subsequently landed on the compromised host.

Additional Data Sources

In addition to ElasticSearch, the raw pilot logs are available here:

http://submit-3.t2.ucsd.edu/CSstoragePath/FactoryLogsGlobalPool/

To use, one needs to know the pilot ID, the factory used to submit the pilot, and the name of the glideinWMS entry point. With this information, one can determine the file name of the pilot log file.

The pilot-based logs are archived (typically within 45 minutes of pilot exit!) using a completely separate mechanism from ElasticSearch. Ideally, having two distinct logging mechanisms - and at least the theoretical possibility of cross-correlating them - will decrease the likelihood of lost data.

This topic: Main > CmsTraceability
Topic revision: r2 - 2017-05-16 - BrianBockelman