Singularity

Brian Bockelman

WLCG Traceability and Isolation WG, 14 Sept 2016

Singularity

- Singularity (http://singularity.lbl.gov) is a project that implements container-based solutions:
    - Launches a payload process inside a given image / environment.
    - Isolates processes in the payload.
    - No UID switching; payload runs as same UID as invoker.
    - Relatively lightweight: no runtime daemon.
    - Features focus on the needs for HPC sites. I.e., original case was for raw disk images sitting on a shared FS. Plays nicely inside a batch system job.
        - It seems the HPC use case is remarkably close to ours!

Under The Hood

- Singularity provides utilities for generating a raw disk image for various popular operating systems.
  - Root access is typically required to generate these.
- It uses a `setuid` binary to chroot, configure
- Further, it has a “contain” mode that provides no external directories by default.

Going forward

- The upcoming 2.2.0 release has quite a few patches / new features for our use case:
  - **Directory-based images**. Instead of a raw disk image, we can point `singularity` toward a directory (such as /cvmfs/cernvm-prod.cern.ch/cvm3).
  - **User-specified bind mounts**. Allows the pilot to control which directories are visible to the payload.
  - **Per-payload scratch directories**. Each job gets a private `/tmp`.

On Security

- Everyone hates `setuid` executables.
    - If you don't, you should!
- I know of no security exploits, but there's a short list of things I'd like to see improved to help protect against future exploits.
- We install Singularity on both our general purpose and our WLCG cluster. I've been through enough of the code that I personally am comfortable with it.
- We have applied to have an independent security audit of the tool.
- It's probably similar size (4.2k lines C, 2.4k lines shell) to the `glexec` core; however, likely much smaller than `glexec+lcmaps+plugins`.
    - No callouts to external services; not linked to external libraries.

Traceability vs Isolation

- Singularity provides *isolation*.
  - Payload cannot attack the pilot or other payloads.
- Singularity does not provide *traceability*.
  - Cannot provide a trusted mapping from a process to an end user.
  - … But can `glexec`? Are you sure that's the binary the user submitted?
- **Editorializing**: It's a Good Thing to have **separate mechanisms for isolation and traceability**. I look forward to tackling these problems separately! No longer need to give up isolation just because you lack a user X509 credential.

Integrating with our ecosystem

- We'll be living the dream if every job was running precisely the same CernVM OS environment!
    - As a sysadmin, I *really really hate* the HEP_OSLibs RPM and never install it.
    - Sysadmins are also free to pick their own OS, independent of experiments.
- There's a pull request for adding native support to HTCondor. Targeting HTCondor 8.7.0 and Singularity 2.2.
    - This would allow glideinWMS-based pools to provide isolation.
    - Features needed for this currently require RHEL7 kernel; if desired, could remove this requirement with modest effort.
- This is something I believe OSG could adopt.
- Singularity 2.2 no longer requires `setuid` on very new kernels; should be plausible to do a self-contained install & config on CVMFS.
    - A few years in the future, and we can isolate by default…

Why Singularity?

- Singularity isolates the filesystem (whitelist writable directories) and the processes (PID namespaces - pilot is ‘invisible’).
- Takes the entire OS environment from CVMFS - even on non-virtualized hosts!
- No UID switching: this reduces the likelihood of confusing the batch system.
- A *roadmap to* `setuid`*-free living*. Version 2.2 no longer needs `setuid` for our use case given a recent kernel.

(Demonstration)