Refactoring GlideinWMS Credentials

Bruno Coimbra, Marco Mambelli, Marco Mascheroni, Namratha Urs, Dennis Box

Throughput Computing 2023

14 July 2023
GlideinWMS

- GlideinWMS is a pilot-based resource provisioning tool for distributed High Throughput Computing
- Provides reliable and uniform virtual clusters
- Leverages HTCondor pools and capabilities
Glideins are pilot jobs that test and customize worker nodes.

They scout and validate the worker node's specifications:
- cores
- memory
- disk
- GPU
- OS
- software
- CVMFS

Customize the worker node's environment:
- libraries
- containers

Enables monitoring and audit.

Runs HTCondor jobs by spawning a startd that joins a virtual cluster.

Handles credentials:
- Pilot credentials (e.g. VO Group)
- Job credentials
GlideinWMS Factory

- **Submits Glideins to sites**
  
  Site details are described in a configuration file
  
  - authentication method    supported VOs    expected resources    …
  
  Only trusted and tested sites are included in production

- HTCondor does the heavy lifting of submissions

- Caches credentials used or forwarded to Glideins
GlideinWMS Frontend

• Requests Glideins to Factories
  
  • Implements a pressure-based system to control the Factory Glidein requests
    Monitors job requests and available entries (sites)
    Keeps a certain number of Glideins running or idle at sites
    Limits requests to enforce policies and avoid overloads

• Manages credentials
  
  Stores generated and VO-provided credentials
  Handles short-term and long-term credential pairs
  Delegates credentials to Factories and Glideins
Traditional X509 Authentication – Pilot Proxy

• A single omniscient proxy is sent from Frontend to Factory to Glidein
• One proxy has multiple purposes
Traditional X509 Authentication – Pilot Proxy

- A single omniscient proxy is sent from Frontend to Factory to Glidein
- One proxy has multiple purposes:
  1. Pilot submission
Traditional X509 Authentication – Pilot Proxy

- A single omniscient proxy is sent from Frontend to Factory to Glidein
- One proxy has multiple purposes:
  1. Pilot submission
  2. Central Manager authentication
Traditional X509 Authentication – Pilot Proxy

- A single omniscient proxy is sent from Frontend to Factory to Glidein
- One proxy has multiple purposes:
  1. Pilot submission
  2. Central Manager authentication
  3. Services access

[Diagram showing the flow of processes from Frontend to Factory to Glidein, including schedd, collector, HTC, Central Manager, and CVMFS.]
Traditional X509 Authentication – Pilot Proxy

- A single omniscient proxy is sent from Frontend to Factory to Glidein
- One proxy has multiple purposes:
  1. Pilot submission (additional credential)
  2. Central Manager authentication
  3. Services access
Traditional X509 Authentication – Pilot Proxy

- A single omniscient proxy is sent from Frontend to Factory to Glidein
- One proxy has multiple purposes:
  1. Pilot submission (additional credential)
  2. Central Manager authentication
  3. Services access
- X509 certificates to identify hosts
Transitioning to Tokens (JWT)

Benefits

- Industry standard
- Ease of use
- Flexible authentication
- Finer grained access control
- Security benefits

Caveats

- Replaces current model
- Requires multiple credentials
- Credentials have many scopes and purposes
- Changes to credential generation and renewal
- Requires a transition plan that supports hybrid systems
Tokens Authentication

- Different credentials serve different purposes
Tokens Authentication

• Different credentials serve different purposes:
  1. Pilot submission (SSH key, SciToken, WLCG token)
Tokens Authentication

- Different credentials serve different purposes:
  1. Pilot submission (SSH key, SciToken, WLCG token)
  2. VO tokens to access resources
Tokens Authentication

- Different credentials serve different purposes:
  1. Pilot submission (SSH key, SciToken, WLCG token)
  2. VO tokens to access resources
  3. CE IDTOKEN for monitoring
Tokens Authentication

- Different credentials serve different purposes:
  1. Pilot submission (SSH key, SciToken, WLCG token)
  2. VO tokens to access resources
  3. CE IDTOKEN for monitoring
  4. Central Manager IDTOKEN
Tokens Authentication

- Different credentials serve different purposes:
  1. Pilot submission (SSH key, SciToken, WLCG token)
  2. VO tokens to access resources
  3. CE IDTOKEN for monitoring
  4. Central Manager IDTOKEN
  5. Job credentials
Tokens Authentication

- Different credentials serve different purposes:
  1. Pilot submission (SSH key, SciToken, WLCG token)
  2. VO tokens to access resources
  3. CE IDTOKEN for monitoring
  4. Central Manager IDTOKEN
  5. Job credentials
- X509 certificates to identify hosts
Credentials Refactoring

• Transitioning from X509 to JWT tokens highlighted the need to refactor the GlideinWMS credentials model

  Our code complexity grew a lot over the years

  Adding support for tokens also added depth and new requirements to credential handling functions

  Other Frontend-like applications such as the HEPCloud Decision Engine need the same changes and features and will benefit from a common library
Credentials Refactoring

- With the refactoring we aim to achieve a few goals:
  1. Credentials handling should be type-agnostic
  2. Credentials should serve specific purposes
  3. Authentication methods should be flexible and predictable
  4. VOs should be able to send an arbitrary number of credentials along with Glideins
  5. VOs should be able to specify additional authentication parameters
  6. Some credentials need to be generated on demand
Generic Credentials

- GlideinWMS will internally handle abstract credentials
- Credentials implement common functions such as:
  - Load from file or text
  - Save to file
  - Decode
  - ...
- Supporting new credential types is done by implementing concrete classes
Credentials Purpose

- Credentials will now serve well defined purposes
  
  ce_authentication  user_pool_authentication  services  ...

- Only credentials intended for CE authentication will be considered when matching Glidein requests to Factory entries

- Credentials for the user pool or services authentication are simply sent along with Glideins
Flexible Authentication Methods

- Factory entries will specify a list of accepted authentication methods
- One authentication method may use multiple credentials
- Multiple authentication methods serve as fallbacks

```python
auth_method="scitoken,grid_proxy;scitoken"
```
VO Credentials List

• VO operators will be able to control which credentials are sent along with Glideins

• By selecting the proper purpose the Frontend will not use the credential but will forward it with to the Glidein

```xml
<security>
  <credentials>
    <credential
      absfname="my.scitoken"
      type="scitoken"
      purpose="ce_auth" />

    <credential
      absfname="my.proxy"
      type="x509_proxy"
      purpose="glidein_payload" />
  </credentials>
</security>
```
VO Parameters List

- Security parameters are now an independent entity
- These parameters are taken in consideration when matching Frontend groups to Factory entries
- Credentials and parameters are considered orthogonally

```xml
<security>
  <credentials> ...
  </credentials>
  <parameters>
    <parameter name="project" value="HEP123" />
    <parameter name="vmid" value="VM123" />
  </parameters>
</security>
```
Credential Generators

- GlideinWMS currently supports SciToken generators
- We will expand generator support to all other credential types

```xml
<credential generator="scitokens_callout"
    absfname="/plugins/scitokens_callout.py"
    security_class="frontend"
    trust_domain="grid"
    type="scitoken"
    purpose="ce_auth"
    comment="generated by osg-token-renewer"
/>
```
Disclaimer

• This work is in active development

• If you have questions, feature requests, or feedback, please get in contact:

  glideinwms-support@fnal.gov

Work in progress!
Acknowledgements

- This work was done under the GlideinWMS project
  
  https://glideinwms.fnal.gov/
  
  https://github.com/glideinWMS/glideinwms

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.