Enabling XSEDE Authentication for Science Gateways
Yu Ma (Indiana University)
Motivation

• XSEDE defines a science gateway as a community-developed set of tools, applications, and data that are integrated via a portal or a suite of applications.

• Science gateways provide a common interface for community users to access computational and data services and resources across CyberInfrastructure (CI).

• Proper user authentication mechanism is crucial:
  – Gateways need to confirm user identity to provide services;
  – Users need to trust gateways for providing their credentials;
  – Service providers need to bookkeeping of user allocations and usages accordingly.
General Approach

- MyProxy is a widely-used open source software for managing security credentials (certificates and keys).
- OAuth for MyProxy (OA4MP) provides an OAuth-compliant REST web interface for certificate delegation:
  - Gateway redirects user to authenticate with a trusted source;
  - Authenticated user provides gateway with a proxy certificate;
  - Gateway in turn accesses CI on user’s behalf using the certificate, hence without directly obtaining user passwords.
- Caveat: Official OAuth for MyProxy client releases require JAVA and Tomcat environment, which may not be easy to deploy for some gateways.
Case Study: NCGAS Galaxy

• National Center for Genome Analysis Support (NCGAS) is funded by NSF and collaborates across multiple institutions (IU, TACC, SDSC and PSC) to provide a variety of computational and data support for biological and genomic researches.
• Galaxy is an open, web-based platform for data intensive biomedical research. The project core is developed in Python.
• NCGAS utilizes Galaxy with customized tools set as a portal for biologists without advanced supercomputing background to access underlying XSEDE resources.
Case Study: NCGAS Galaxy

NATIONAL CENTER FOR GENOME ANALYSIS SUPPORT
INDIANA UNIVERSITY

Welcome to the Galaxy instance at National Center for Genome Analysis Support

Thank you for choosing Galaxy! NCGAS is committed to providing support for NSF funded genome research. Don't hesitate to contact help@ncgas.org if you find that you need a tool that is not supported by our current Galaxy or if you have questions or suggestions. When possible, we will have the requested tool up and running in two working days time, and failing that, will report the status of the request within that time.

This instance of the Galaxy is installed and maintained by National Center for Genome Analysis Support NCGAS (NSF Award #1062432).

The computing power is provided by the Indiana University Mason Compute Cluster

The storage is provided by the Indiana University Data Capacitor (NSF Award # 0521433)

The web server is hosted on the Indiana University Quarry Gateway Hosting.

The Galaxy project is supported in part by NSF, NIGRI, and the Huck Institutes of the Life Sciences.

NCGAS is a center of the Pervasive Technology Institute at Indiana University

Questions? help@ncgas.org

NCGAS © 2014 | National Center for Genome Analysis Support | Pervasive Technology Institute
Authentication for NCGAS Galaxy

• NCGAS Galaxy preferred user authentication requires:
  – XSEDE users can authenticate with NCGAS Galaxy through InCommon credentials.
  – Only NCGAS authorized users can authenticate and use the resource.

• Galaxy uses built-in HTTP server in Python, and recommends proxy through Apache or Nginx in production environment.

• Galaxy supports three authentication mechanisms:
  – Internal user accounts using email address and password;
  – OpenID authentication with Galaxy as a relying party;
  – HTTP REMOTE_USER provided by front-end web server;
InCommon Credential Delegation: CILogon

- CILogon Service (http://www.cilogon.org) supports MyProxy OAuth protocol for certificate delegation to web applications, allowing users to authenticate with their home organization.

Image source: http://www.cilogon.org/portal-delegation
Technical Challenges

• CILogon client customizes OAuth for MyProxy client, hence requires Java and Tomcat, while Galaxy prefers Python and Apache/Nginx.
• Python lacks full featured OAuth libraries supporting RSA-SHA1 signature method required by CILogon's OAuth interface.
• Once authenticated through CILogon, remote username needs to be forwarded to Galaxy via Apache proxy;
• Additional authorization is required for CILogon authenticated users;
• Some default CILogon identity providers including OpenID providers (Google, PayPal, Verisign) are not allowed by NCGAS.
NCGAS Galaxy Authentication Architecture

Authentication
- Apache Web Server
- HTTP_COOKIE
- PHP CILogon OAuth Client

CILogon

Galaxy
Technical Highlights

• PHP (non Java) implementation of CILogon OAuth Client.
• Configure Apache proxy to Galaxy:
  – Enable Galaxy external user authentication (universe_wsgi.ini);
  – Configure Apache for proxy forwarding; (httpd ssl.conf);
  – Configure Apache for CILogon authentication with HTTP_COOKIE rewrite; (httpd ssl.conf)
• Customized NCGAS Skin limiting IdP to InCommon academic.
• PHP implementation of simple file-based user authorization.
• Lightweight, packaged for general Galaxy installation.
• Easy to deploy and in production at https://galaxy.ncgas.org.
• Open source and more details at: https://github.com/SciGaP/Galaxy-CILogon-Authentication
NCGAS Galaxy Authentication in Production
Case Study: GAAMP

• General Automated Atomic Model Parameterization (GAAMP) is developed by Roux Lab at University of Chicago to provide a range of tools and services for the construction, parameterization, and optimization of empirical atomic force fields used in biomolecular computations.

• GAAMP XSEDE gateway is a lightweight portal developed in Python CGI to help XSEDE users utilize GAAMP across XSEDE computing resources.
GAAMP XSEDE Gateway

General Automated Atomic Model Parameterization

Membrane Protein Structural Dynamics Gateway

Parameterization for amino acid ONLY (tutorial)

Please make sure GAAMP gateway can access your server correctly at here if you need the job submitted automatically. You can submit the job by yourself.

Submit your job to server
Your account name on the server
Your email to receive results
Molecular coordinate (PDB/AMOL2)
The net charge of your compound
Index of atom L to be deleted
L/D configuration for C-alpha
Initial force field based on

Questions? E-mail: gaamp.uchicago@gmail.com
© 2011 Consortium for Membrane Protein Dynamics
Authentication for GAAMP XSEDE Gateway

• GAAMP gateway currently lacks proper XSEDE authentication mechanism:
  – Collect XSEDE user account information via web form;
  – Generate and package necessary files for user to download;
  – User login XSEDE supercomputer (trestles.sdsc.edu or blacklight.psc.teragrid.org) to submit jobs.

• GAAMP XSEDE gateway user authentication requires:
  – XSEDE user can authenticate with GAAMP gateway.
  – GAAMP gateway can submit jobs on user’s behalf, hence automating the current job submission workflow.
Approaches

• XSEDE OAuth ([https://www.xsede.org/oauth/](https://www.xsede.org/oauth/)) supports OA4MP delegation services for gateways.

• Adapt NCGAS CILogon PHP client for XSEDE OAuth:
  – Modify OAuth Endpoints (4 lines).
  – Save the user proxy certificate to file instead of discarding after the username is extracted (4 lines).

• Upon authentication, redirect to GAAMP gateway and forward user proxy certificate file information via cookie in HTTP request.

• GAAMP gateway can submit jobs to XSEDE resources via GSI-SSH on user’s behalf using the proxy certificate.
GAAMP XSEDE Authentication Architecture

PHP OA4MP Client

HTTP COOKIE
X509 User Proxy

XSEDE OAuth

PSU Blacklight
SDSC Trestles

GSI-SSH
Technical Highlights

- PHP implementation of XSEDE OAuth client.
- Generic XSEDE authentication solution and easy to adopt for other gateways.
- Under deployment for production GAAMP gateway.
- Open source and more details at: https://github.com/SciGaP/ECSS-GAAMP-Gateway
Outcomes

- XSEDE authentication solution for two projects:
  - NCGAS Galaxy: in production and follow up with XSEDE Galaxy efforts (led by Philip Blood).
  - GAAMP: under deployment for production.

- Code nuggets available for easy adoption:
  - Galaxy gateways: https://github.com/SciGaP/Galaxy-ClLogon-Authentication

- Need more help? Contact us: auss-gateways@xsede.org
Software Nuggets derived from ECSS Projects

ECSS Symposium – March 18th 2014

XSEDE
Extreme Science and Engineering Discovery Environment
Outline

• Package software utility libraries from ECSS Gateway Projects
• Job Management for Science Gateways
  – Job Submission
  – Job Monitoring
• Source code
• Questions & Discussion
• Contact auss-gateways@xsede.org
Gateway Job Management Overview

- **End Users**
- **Science Gateway Middleware**
- **Portals/Desktop Clients**
- **Java CoG Abstraction**
- **JSDL/BES Abstraction**
- **GSI-SSH**
- **GRAM**
- **Unicore**
- **Local Resource Managers**

**XSEDE Service Providers**
GSI-SSH Gateway Job submission

• GridChem
  – Fork GSI-SSH/SCP commands to invoked Unified_Scripts on remote machine which create and submit the PBS/Batch jobs.

• CyberGIS/GISolve/SimpleGrid
  – Create the PBS options on command line and the qsub and the command options

• CIPRES
  – Prepare the PBS script and use Java Process Builder to fork of ssh commands
Motivation of a programmable SSH based job submission library

• gsi-ssh commands from gateway machines required installation of grid clients.
• Forking commands looses control from calling program. A library will address this.
• PBS/SLURM Job submission scripts can be templated.
• Create a remote gsi-ssh light weight API targeted for gateway developers.
Existing Programmatic GSI-SSH Examples

- GSI-SSHTerm
  - Terminal application
  - Lot of UI code
  - Integrated with XSEDE
  - Uses J2SSH & cog-jglobus
  - Not easy to refactor as an API due to UI code
  - UK E-science created and maintained it
  - But no active maintenance now
Eclipse P2P Project

- Bundles a forked version of Jcraft SSH code and customized to do GSI-SSH operations
- Uses Jglobus for Myproxy interactions
- Builds over NCSA Lcrypto Library
- Very similar to GSI-SSHterm usage
Goals of the Library

• Borrow ideas from GSI-SSH term and Eclipse P2P and develop a distributable library
• Port to Jglobus 2.0
• Extend widely used and publicly available JcraL code instead of modifying or forking.
• Have working examples so new gateways using SSH can easily port to using GSI-SSH
• Using different authentication mechanisms
• Keep it simple
Job Submission API

- PBS Script (Trestles)
- SLURM Script (Stampede)
- Other clusters as needed
- SSH Utilities and output parsing
- Generic Command Execution
- GSISSH Authentication
- SSH Authentication
Key API Features

• Create working directories
• Create Batch Scripts
• Submission Jobs
• Monitor Jobs
• List Outputs
Example Submission for Illustration (will provide download links)

```java
// authentication information for SSH either with username password or ssh keys
AuthenticationInfo authenticationInfo = null;
if (password != null) {
    authenticationInfo = new DefaultPasswordAuthenticationInfo(this.password);
} else {
    new DefaultPublicKeyFileAuthentication(this.publicKeyPath, this.privateKeyPath,
                                             this.passPhrase);
}

// Server info
ServerInfo serverInfo = new ServerInfo(this.userName, this.hostName);
Cluster pbsCluster = new PBSCluster(serverInfo, authenticationInfo, CommonUtils.getPBSJobManager("/opt/torque/torque-4.2.3.1/bin"));

// constructing the job object
JobDescriptor jobDescriptor = new JobDescriptor();
jobDescriptor.setWorkingDirectory(workingDirectory);
jobDescriptor.setShellName("/bin/bash");
jobDescriptor.setJobName("GSI_SSH_SLEEP_JOB");
jobDescriptor.setExecutablePath("/bin/echo");
jobDescriptor.setAllEnvExport(true);
jobDescriptor.setMailOptions("n");
jobDescriptor.setStandardOutFile(workingDirectory + File.separator + "application.out");
jobDescriptor.setStandardErrorFile(workingDirectory + File.separator + "application.err");
jobDescriptor.setNodes(1);
jobDescriptor.setProcessesPerNode(1);
jobDescriptor.setQueueName("normal");
jobDescriptor.setMaxWallTime("5");
jobDescriptor.setJobSubmitter("aprun -n 1");
List<String> inputs = new ArrayList<>();
jobDescriptor.setInputValues(inputs);

// finished construction of job object
System.out.println(jobDescriptor.toXML());
String jobID = pbsCluster.submitBatchJob(jobDescriptor);
System.out.println("JobID returned : " + jobID);
```
Download

• Source Code – GIT Location
  – https://github.com/apache/airavata/tree/master/tools/gsissh
  – Maven Build
  – mvn clean install -Dmyproxy.user=ogce -Dmyproxy.password=xxxx

• The jar is available in Maven Central
  – Search for gsissh
  – http://search.maven.org/#search%7Cga%7C1%7Cgsissh
Job Monitoring
Job Monitoring tool

• Push Monitor
  – Integrate with AMQP based monitoring for XSEDE resources.

• Pull Monitor
  – Using gsissh tool to poll the Job Manager
  – Each command uses different output parser
  – Supports qstat and squeue monitoring
    • Not ideal: frequent polling, latencies with polling frequencies.
    • But reliable – using as a fall-back
Simplified view of XSEDE AMQP Service

- XSEDE User Portal
- Science Gateways
- GLUE2 Documents (JSON)
- Information Publishing Framework
- Cluster Scheduler
- XSEDE Service Provider (Stampede)
- XSEDE SP: Blacklight

Slide Source: Warren Smith
Job Monitoring Summary

- Job Manager
- Pull Monitor Queue
- Push Monitor Queue
- AMQP Monitor
- Qstat Monitor
- Job status Event Bus
- Logging Listener
- Registry Listener
- ...
// initialize monitor component with AMQP Monitor
monitorManager = new MonitorManager();
AMQPMonitor amqpMonitor = new
  AMQPMonitor(monitorManager, getMonitorPublisher(),
  monitorManager.getPullQueue(), monitorManager.getFinishQueue(),
  "Users/lahirugunathilake/Downloads/x509up_u503876", "xsede_private",
  Arrays.asList("info1.dyn.teragrid.org", info2.dyn.teragrid.org".split("","")));

monitorManager.addPushMonitor(amqpMonitor);
monitorManager.launchMonitor();

/* now have to submit a job to some machine and add that job to the queue */
//Create authentication
GSIAuthenticationInfo authenticationInfo
  = new MyProxyAuthenticationInfo(myProxyUserName, myProxyPassword, "myproxy.teragrid.org",
  7512, 17280000, certificateLocation);

// Server info
ServerInfo serverInfo = new ServerInfo("ogce", "trestles.sdsc.edu");

Cluster pbsCluster = new
  PBSCluster(serverInfo, authenticationInfo, org.apache.airavata.gsi.ssh.util.CommonUtils.getPBSJobManager("/opt/torque/bin/"));

System.out.println("Target PBS file path: " + workingDirectory);
// constructing the job object
JobDescriptor jobDescriptor = new JobDescriptor();
jobDescriptor.setWorkingDirectory(workingDirectory);
jobDescriptor.setShellName("/bin/bash");
jobDescriptor.setJobName("GSI_SSH_SLEEP_JOB");
jobDescriptor.setExecutablePath("/bin/echo");
jobDescriptor.setAllEnvExport(true);
jobDescriptor.setMailOptions("n");
jobDescriptor.setStandardOutFile(workingDirectory + File.separator + "application.out");
jobDescriptor.setStandardErrorFile(workingDirectory + File.separator + "application.err");
jobDescriptor.setNodes(1);
jobDescriptor.setProcessesPerNode(1);
jobDescriptor.setQueueName("normal");
jobDescriptor.setMaxWallTime("60");
jobDescriptor.setAccountString("sds128");
List<String> inputs = new ArrayList<>();
jobDescriptor.setOwner("ogce");
inputs.add("Hello World");
jobDescriptor.setInputValues(inputs);
//Finished construction of job object
String jobID = pbsCluster.submitBatchJob(jobDescriptor);

// Adding job to the Queue to monitor
monitorManager.addAJobToMonitor(new MonitorID(hostDescription, jobID, null, null, "ogce"));
Summary: Tools for Gateway Job Submission & Monitoring

Portals/Desktop Clients

Science Gateway Middleware

Job Submission Abstraction

File transfer

Listener

Job Monitoring

File transfer

XSEDE Service Providers

Local Resource Managers

GRAM

GSI-SSH

Unicore

XSEDE AMQP Message Bus

Push Monitoring

Pull Monitoring

Publish

GSI-SSH

HTML5/AJAX MQ Clients

Mobile Listeners
Download

• Source Code – GIT Location
  – Maven Build
  – mvn clean install -Dmyproxy.user=ogce -Dmyproxy.password=xxxx
Contact & Future Work

• Future Work:
  – Improving documentation and samples.
  – Make the tools more standalone by minimizing external dependencies.

• Contact:
  – If you have a ECSS project you would like to use these tools, contact Auss-gateways@xsede.org
Our reach will forever exceed our grasp, but, in stretching our horizon, we forever improve our world.