



# Credential Management in HTCondor

Derek Weitzel
(Based on slides by Zach Miller)

# Talking about HTCondor

- I'm going to be talking about HTCondor's integration with Tokens (SciTokens and others)

- If you want a detailed talk about SciTokens, see Brian's talk at the Naples workshop

# Overview

- Jobs often need to access data or services while they are running
- These services often require authorization
- Authorization often requires proving identity
- Examples:
  - x509 Certificate/proxy
  - Username/password
  - Tokens/capabilities

# Overview

- Jobs may also want to store their output on a server somewhere instead of bringing it back to the submit machine
- Servers typically require some kind of authorization to write data to them.
- Examples:
  - box.com, Dropbox, Google Drive
  - AFS home directory

# Existing Support

- HTCondor has long supported GSI
- These are x509 certificates and proxies
- The proxy travels along with the job
- Any other credentials can be moved using HTCondor's file transfer mechanism
  - MAKE SURE THE TRANSFER IS SECURE!
  - encrypt_input_files

# Generalizing

- Credentials should not be treated as regular files
- Need to be moved securely
- May need to be refreshed during the job
- May wish to limit their abilities (make them shorter lived, limited in capability, etc.)
- Might need to use them to retrieve actual input data
  - Which means they need to be handled first

# The Components

- **condor_credd**

- Runs alongside the condor_schedd
- Manages credentials on the submit machine
- Has a “private” (root-owned and protected) directory on submit machine
  - SEC_CREDENTIAL_DIRECTORY

# The Components

- **condor_credd**
- Views credentials as opaque objects
- Other than creating and destroying, is not able to manipulate the credentials

# The Components

- **Credential Monitor**

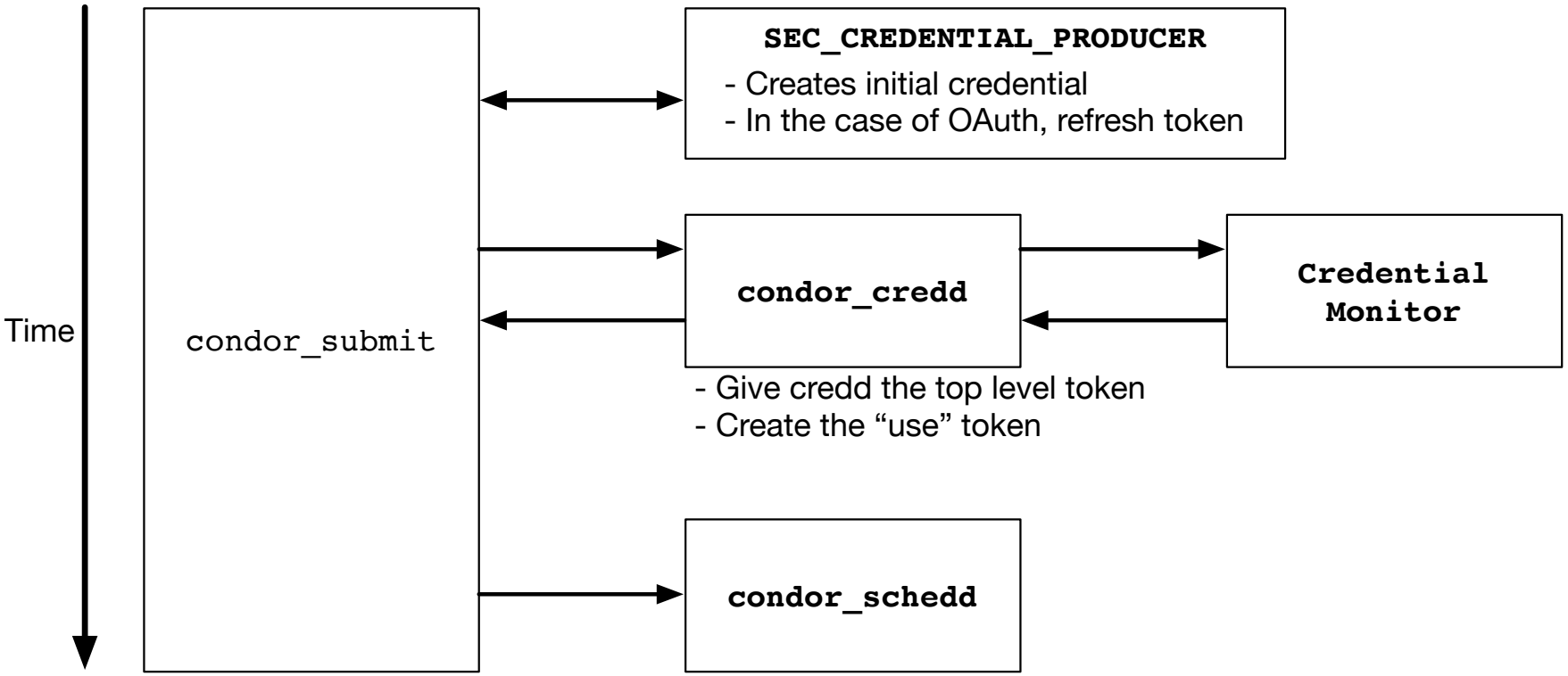
- Supplied by the site administrator (not the HTCondor developers)
- Aware of how to manipulate and refresh credentials
- Monitors and transforms files in the SEC_CREDENTIAL_DIRECTORY

# The Components

- **SEC_CREDENTIAL_PRODUCER**

- Script or executable provided by site administrator
- Runs on behalf of user to generate a credential that is given to the condor_credd
- Storing credential must succeed before job submission will continue

# Submit Process



# Credential Monitor

- Two types of credentials
  - "Top" level
  - "Use" level

- The SEC_CREDENTIAL_PRODUCER supplies the "top" level
- The Credential Monitor converts the "top" into a "use" credential

# OAuth / SciTokens

- The “top” credential is a refresh credential. This is presented to a token server to get “use” credentials which are access tokens
- Only the access tokens are shipped to the execute machine
- Periodically refreshed by pulling from the submit machine

# OAuth

- Here, the user needs to obtain their “top” (refresh) tokens before the job can be submitted
- `condor_submit` returns a URL to the user if they lack necessary tokens.
- The OAuth Credential Monitor hosts a web interface at this URL to assist the user in this, receives the refresh tokens, and uses them to keep copies of fresh access tokens

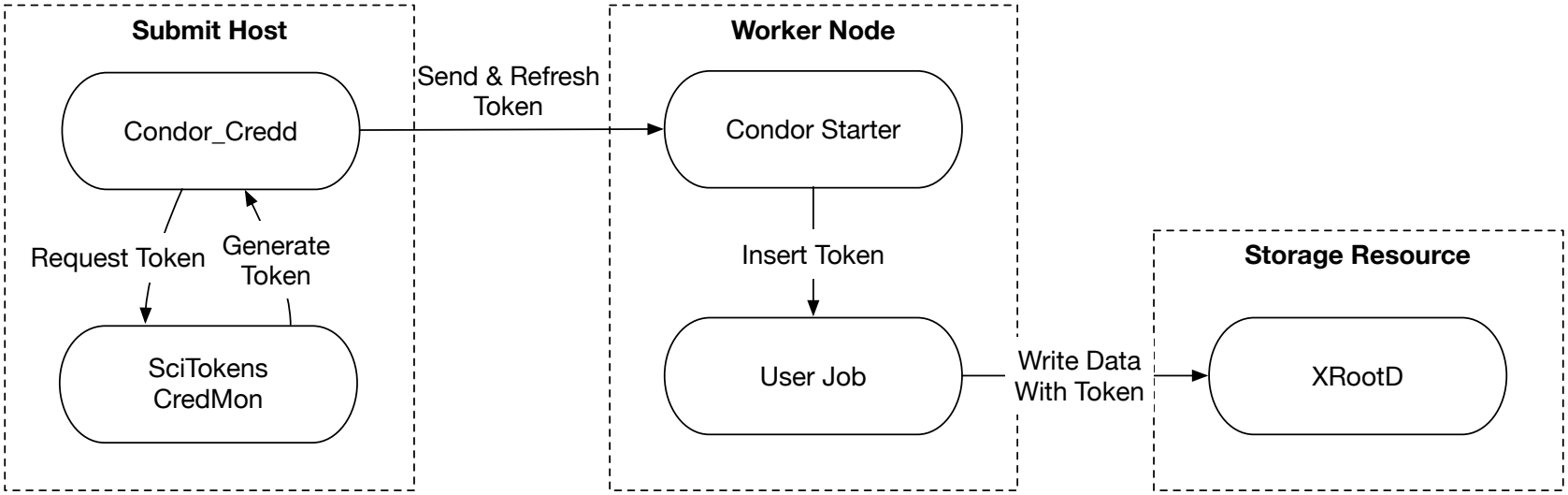
# OAuth

- The job can then use its tokens to get/put data, such as contacting a server (box.com, Google Drive, XRootD, stashcache)
- Again, only the access tokens are sent with the job
- No need for a Credential Monitor on the execute side.
- This keeps users' identity credentials off all the execute machines

# Job Execution

- When a user has no jobs left running on a machine, their creds are marked for removal
- A timer periodically causes all marked credentials to be removed
- Likewise, when a user has no jobs left in the job queue, their creds are marked and periodically removed.

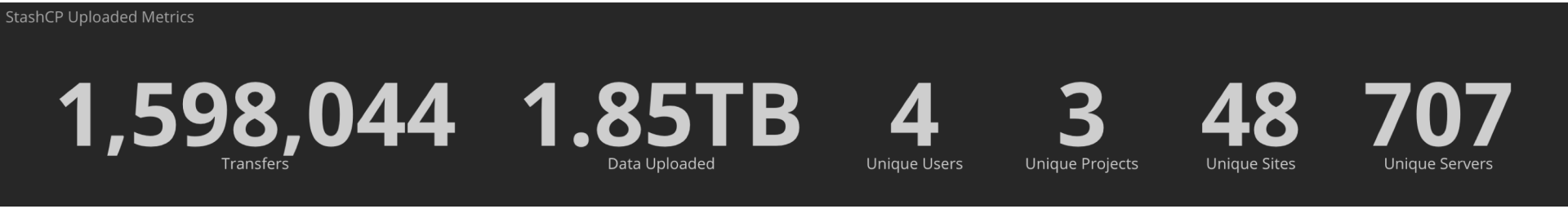
# SciTokens




1. Token is created on demand by the SciTokens Credmon.
2. Token is copied to the Shadow and then to the user's job environment
3. Transfer tools use the token to authenticate with the storage resources
4. Storage resource trusts the cryptographically signed token from the user's job, without any lookup required.

# SciTokens

- This is already getting used in the OSG
- Bioinformatics group at Clemson is using SciTokens to send data to an XRootD server
- This in turn writes the data into the “Stash” storage at the University of Chicago



# Summary

- The same general HTCondor components and architecture supports that, more general OAuth support, as well as Kerberos support
- Site-specific pieces can be provided by the local admin

# Summary

- Using customizable pieces allows for flexibility and also expansion of the credential management system
- New types of credential support can be added without HTCondor code changes

# Summary

- Still a work in progress…
  - For v8.8 things will be documented
  - Reference versions of the Credential Monitor for OAuth and Kerberos will be supplied with the HTCondor release
- Many thanks to the folks at CERN and DESY for working through this initial implementations