HTCondor Security Mechanisms Overview

HTCondor Security

- Allows authentication of users and daemons
- Encryption over the network
- Integrity checking over the network

"locks-masterlocks.jpg" by Brian De Smet, © 2005
Used with permission.
http://www.fief.org/sysadmin/blosxom.cgi/2005/07/21#locks

Authorization

- HTCondor users ALLOW / DENY lists to control authorization
- There are different levels of access in HTCondor, and each can have a separate authorization list and security policy.

Authorization

- Possible values for authorization levels:
  - CLIENT
  - READ
  - WRITE
  - CONFIG
  - ADMINISTRATOR
  - OWNER
  - DAEMON
  - NEGOTIATOR

Authorization Levels

- READ
  - querying information
  - **`condor_status`**, **`condor_q`**, etc
- WRITE
  - updating information
  - **`condor_submit`**, adding nodes to a pool, sending ClassAds to the collector, etc
  - Includes READ

Authorization Levels

- ADMINISTRATOR
  - Administrative commands
  - **`condor_on`**, **`condor_off`**, **`condor_reconfig`**, **`condor_restart`**, etc.
  - Includes READ and WRITE

Authorization Levels

- DAEMON
  - Daemon to daemon communications
  - Includes READ and WRITE
- NEGOTIATOR
  - **`condor_negotiator`** to other daemons
  - Includes READ

Authorization

The full hierarchy of authorization levels:

$$\begin{tikzcd}
\text{ADMINISTRATOR} \arrow[dr] & \text{DAEMON} \arrow[d] & \text{NEGOTIATOR} \arrow[ddl] \\
\text{OWNER} & \text{WRITE} \arrow[d] & \text{CONFIG} \arrow[dl] \\
\text{CLIENT} & \text{READ} &
\end{tikzcd}$$

Authorization

› There is a separate ALLOW / DENY list for each authorization level.
› DENY takes preference over ALLOW

```
ALLOW_READ = *
ALLOW_WRITE = *.cs.wisc.edu
DENY_WRITE = zeroday.cs.wisc.edu
ALLOW_ADMINISTRATOR = condor.cs.wisc.edu
```

Host-based Authorization

› More Examples:

```
ALLOW_WRITE = *
ALLOW_WRITE = goose.cs.wisc.edu
ALLOW_WRITE = *.cs.wisc.edu
ALLOW_WRITE = 128.105.*
ALLOW_WRITE = 128.105.0.0/16
```

Host-based Authorization

- Each entry is a comma-separated list.
- Wildcards are allowed only at the beginning of hostnames or at the end of IP addresses.
- Subnets are supported using a / and number of significant bits.

```
HOSTALLOW_WRITE = *.cs.wisc.edu, *.engr.wisc.edu

HOSTALLOW_WRITE = 128.105.*, *.engr.wisc.edu, 128.105.64.0/18
```

Host-based Authorization

- This is the default setup, which has some shortcomings but is easy to configure.
- Allows you to specify capabilities by hostname, IP address, and/or subnet.

Problems With Default Installation

› Host-based granularity is too big
  - Any user who can login to central manager has “Administrator” privileges

```
HOSTALLOW_ADMINISTRATOR = $(CONDOR_HOST)
```

  - Any user on an execute machine can evict the job on that machine via condor_vacate

```
HOSTALLOW_OWNER = $(FULL_HOSTNAME)
```

Problems With Default Installation

- Most connections are NOT authenticated
  - Queue management commands (condor_submit, condor_hold, etc.) are because Condor explicitly forces authentication.
  - Daemon-to-daemon commands are not.
  - It is possible to send false information to the collector and other denials of service

Problems With Default Installation

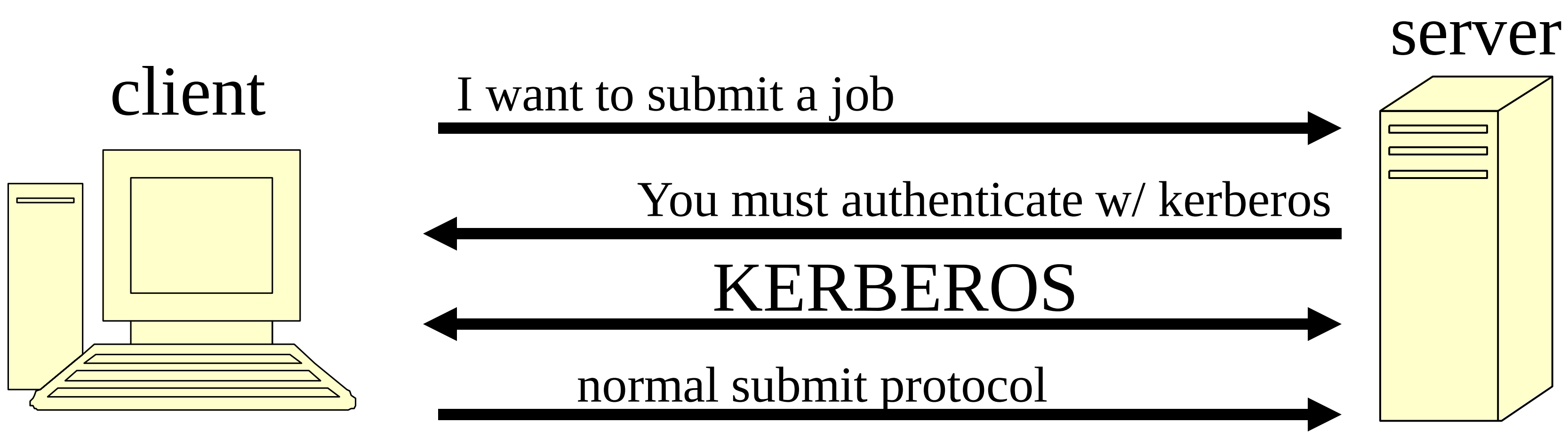
- Traffic is not encrypted or checked for integrity.
  - Possibility of someone eavesdropping on your traffic, including files transferred to or from execute machine
  - Possibility of someone modifying your traffic without detection

Security Policy Configuration

- Condor provides many mechanisms to address the previous shortcomings:
  - Many authentication methods
  - Strong encryption
  - Signed checksums for integrity

Security Policy Configuration

› Condor will negotiate security requirements and supported methods



Security Policy Configuration

Default Policy

```
SEC_DEFAULT_ENCRYPTION = OPTIONAL
SEC_DEFAULT_INTEGRITY = OPTIONAL
SEC_DEFAULT_AUTHENTICATION = OPTIONAL
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, GSI, KERBEROS, SSL, PASSWORD       #UNIX
SEC_DEFAULT_AUTHENTICATION_METHODS = NTSSPI, KERBEROS, SSL, PASSWORD        #WIN32
```

Security Policy Configuration

Default Policy

```
SEC_DEFAULT_ENCRYPTION = OPTIONAL
SEC_DEFAULT_INTEGRITY = OPTIONAL
SEC_DEFAULT_AUTHENTICATION = OPTIONAL
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, GSI, KERBEROS, SSL, PASSWORD        #UNIX
SEC_DEFAULT_AUTHENTICATION_METHODS = NTSSPI, KERBEROS, SSL, PASSWORD         #WIN32
```

Possible Policy Values

NEVER	do not allow this to happen
OPTIONAL	do not request it, but allow it
PREFFERED	request it, but do not require it
REQUIRED	this is mandatory

Security Policy Configuration

Policy Reconciliation

Client Policy	Server Policy: R	P	O	N
Required	Y	Y	Y	X
Preferred	Y	Y	Y	N
Optional	Y	Y	N	N
Never	X	N	N	N

Security Policy Configuration

Policy Reconciliation Example

CLIENT POLICY

```
SEC_DEFAULT_ENCRYPTION = OPTIONAL
SEC_DEFAULT_INTEGRITY = OPTIONAL
SEC_DEFAULT_AUTHENTICATION = OPTIONAL
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, GSI, KERBEROS, SSL, PASSWORD
```

SERVER POLICY

```
SEC_DEFAULT_ENCRYPTION = REQUIRED
SEC_DEFAULT_INTEGRITY = REQUIRED
SEC_DEFAULT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION_METHODS = SSL
```

RECONCILED POLICY

```
ENCRYPTION = YES
INTEGRITY = YES
AUTHENTICATION = YES
METHODS = SSL
```

Security Policy Configuration

Once you have authenticated users, you may use a more fine-grained authorization list:

```
ALLOW_WRITE = zmiller@cs.wisc.edu

ALLOW_WRITE = zmiller@cs.wisc.edu/goose.cs.wisc.edu

ALLOW_WRITE = zmiller@cs.wisc.edu/*.cs.wisc.edu
```

Security Policy Configuration

- Format of canonical username:

  user@domain/host
- One wildcard allowed in the user@domain portion, and one allowed in the host portion
- If there is no '/' character, Condor will do one of two things:
  - If there is an '@' character, it is assumed to be a username, and maps to user@domain/*
  - If there is no '@', it is assumed to be a hostname and maps to */hostname

Example Policies

› Allow anyone from wisc.edu:

```
ALLOW_READ=*.wisc.edu
```

› Allow any authenticated local user:

```
ALLOW_READ=*@wisc.edu/*.wisc.edu
```

› Allow specific user/machine

```
ALLOW_NEGOTIATOR= \
  daemon@wisc.edu/condor.wisc.edu
```

AUTHENTICATION_METHODS

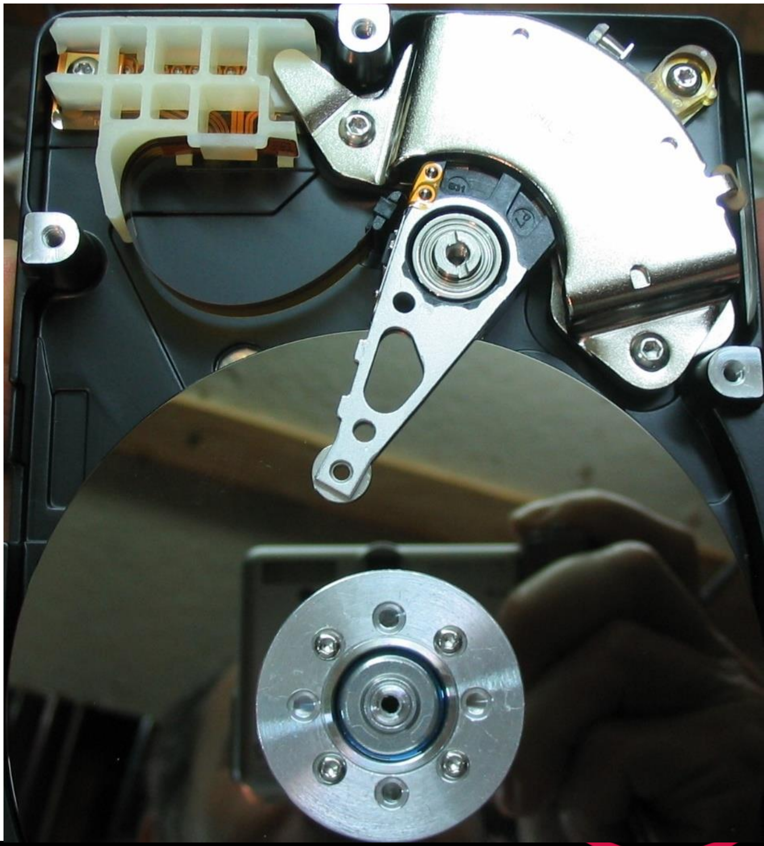
- How to authenticate users and daemons?
  - NTSSPI – Microsoft Windows
  - FS – (UNIX) Local file system
  - FS_REMOTE – (UNIX) Network file system
  - CLAIMTOBE – Insecure, for testing
  - ANONYMOUS – Insecure
  - PASSWORD – Shared secret
  - SSL – Public key encryption
  - Kerberos – Requires existing KDC setup
  - GSI – Globus/Grid Security Infrastructure

NTSSPI
Microsoft Windows

- Only works on Windows
- Password must be the same on both systems
- No configuration required

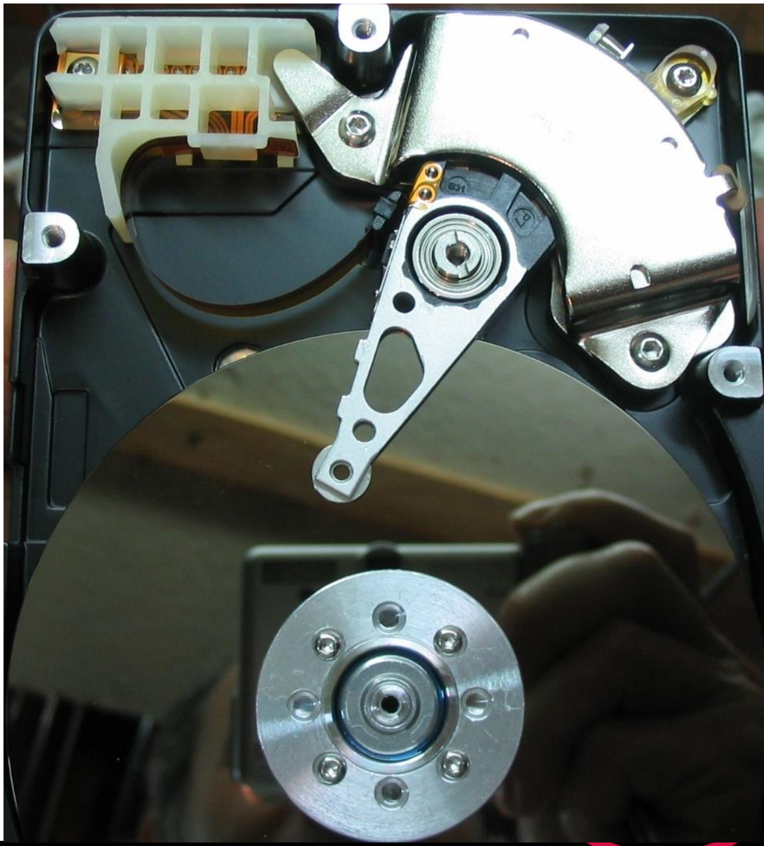
FS: File System

- Checks that the user can create a directory owned by the user.
  - Only works on local machine (uses /tmp)
  - Assumes filesystem is trustworthy
- No configuration required



FS_REMOTE

- Checks that the user can create a directory owned by the user on a shared filesystem
  - Works across machines
  - Assumes filesystem is trustworthy!!! THIS IS NOT ALWAYS TRUE!
  - Target directory must be properly configured.



CLAIMTOBE

› CLAIMTOBE - Just what it sounds like
  - Allows client to send any ID
  - Very insecure
  - Useful for testing

PASSWORD

- Shared secret
- Only suitable for daemon-to-daemon communications, not for authenticating end users
- Always authenticates as principle “condor_pool@$(UID_DOMAIN)”
- Simple
- Works on both UNIX (using filesystem protection) and Windows (using secure registry storage)

SSL

- Public key encryption system
- Daemons and users have X.509 certificates
- Flexible – all Condor daemons in pool can share one certificate, or use one cert per host.
- Map file transforms X.509 distinguished name into an identity (see later slides on mapping)

Kerberos and GSI

- Complex to set up
- Useful if you already use one of these systems
- The most secure methods HTCondor provides (along with SSL)

Security Policy Configuration

- Map file controls how credentials are mapped to HTCondor user principals.
- In your condor_config:

```
CERTIFICATE_MAPFILE = /path/to/mapfile
```

- Each line is a mapping rule.
- Each rule has three fields:

  method   regex   mapped_name

  (any field with spaces should be quoted)

Security Policy Configuration

› Some example map file entries:

(These should be one line, they are split here)

```
SSL
    "/C=US/ST=Wisconsin/L=Madison/O=University of Wisconsin -- Madison/O=Computer
    Sciences Department/OU=Condor Project/CN=Zach Miller/Email=zmiller@cs.wisc.edu"
          zmiller@cs.wisc.edu

SSL
    "/C=US/ST=Wisconsin/L=Madison/O=University of Wisconsin -- Madison/O=Computer
    Sciences Department/OU=Condor Project/CN=Todd Tannenbaum/Email=tannenba@cs.wisc.edu"
          tannenba@cs.wisc.edu

Etc.
```

Security Policy Configuration

› Example with Regular Expression:
- RegEx matches and sub-matches can be referenced using \1, \2, etc.
- The map file gives you a canonical name from the authenticated user:

```
SSL          Email=(.*)    \1
```

```
"/C=US/ST=Wisconsin/L=Madison/O=University of Wisconsin –
Madison/O=Computer Sciences Department/OU=Condor Project
/CN=Zach Miller/Email=zmiller@cs.wisc.edu"
```

$\longrightarrow$ zmiller@cs.wisc.edu

Security Policy Configuration

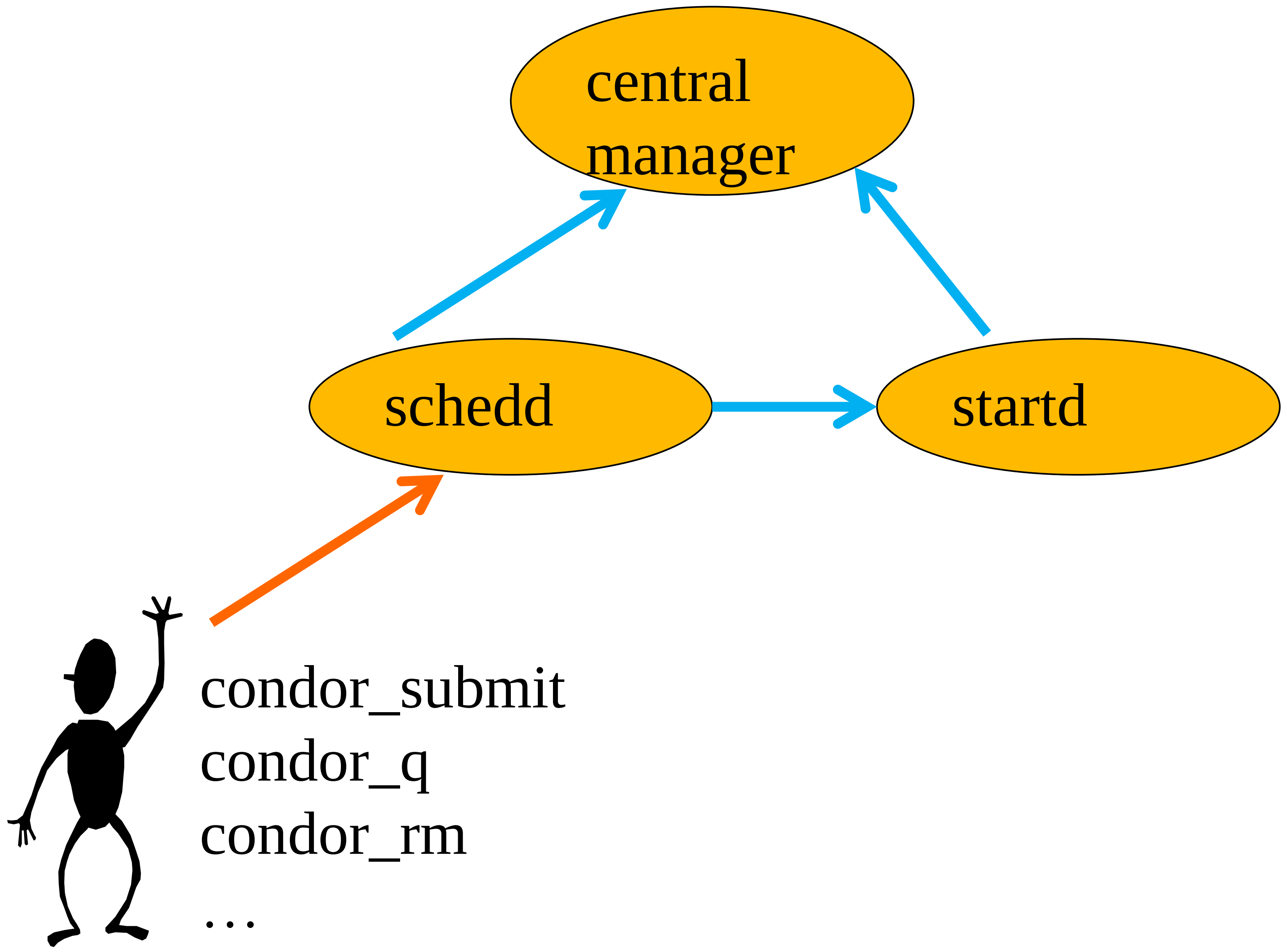
› Default map file:
  (each line is <method> <regex> <mappedname>)

```
FS (.*) \1
FS_REMOTE (.*) \1
GSI (.*) GSS_ASSIST_GRIDMAP
SSL (.*) unmapped
KERBEROS (.*) \1
NTSSPI (.*) \1
CLAIMTOBE (.*) \1
ANONYMOUS (.*) CONDOR_ANONYMOUS
PASSWORD (.*) \1
```

(Special Token to call Globus)

Example Security Configuration

- Let's put it all together with an example.
- Desired policy, in English:
  - Authenticate, encrypt, and do integrity checks on everything.
  - Use SSL authentication for daemon-to-daemon communication
  - Use FS (or SSL) authentication for users so that we don't need to issue certs to everyone.


central manager
schedd
startd
condor_submit
condor_q
condor_rm
…

Example Security Configuration

```
# Turn on all security options:
SEC_DEFAULT_AUTHENTICATION=REQUIRED
SEC_DEFAULT_ENCRYPTION=REQUIRED
SEC_DEFAULT_INTEGRITY=REQUIRED
```

Example Security Configuration

```
# Specify allowed methods:
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, SSL
SEC_DAEMON_AUTHENTICATION_METHODS = SSL
```

› Requires giving your daemons an X.509 certificates
› You will also need a map file for SSL distinguished names. Let’s assume the daemon cert maps to daemon@wisc.edu.
› Let’s also assume the admin has a cert that maps to admin@wisc.edu

Example Security Configuration

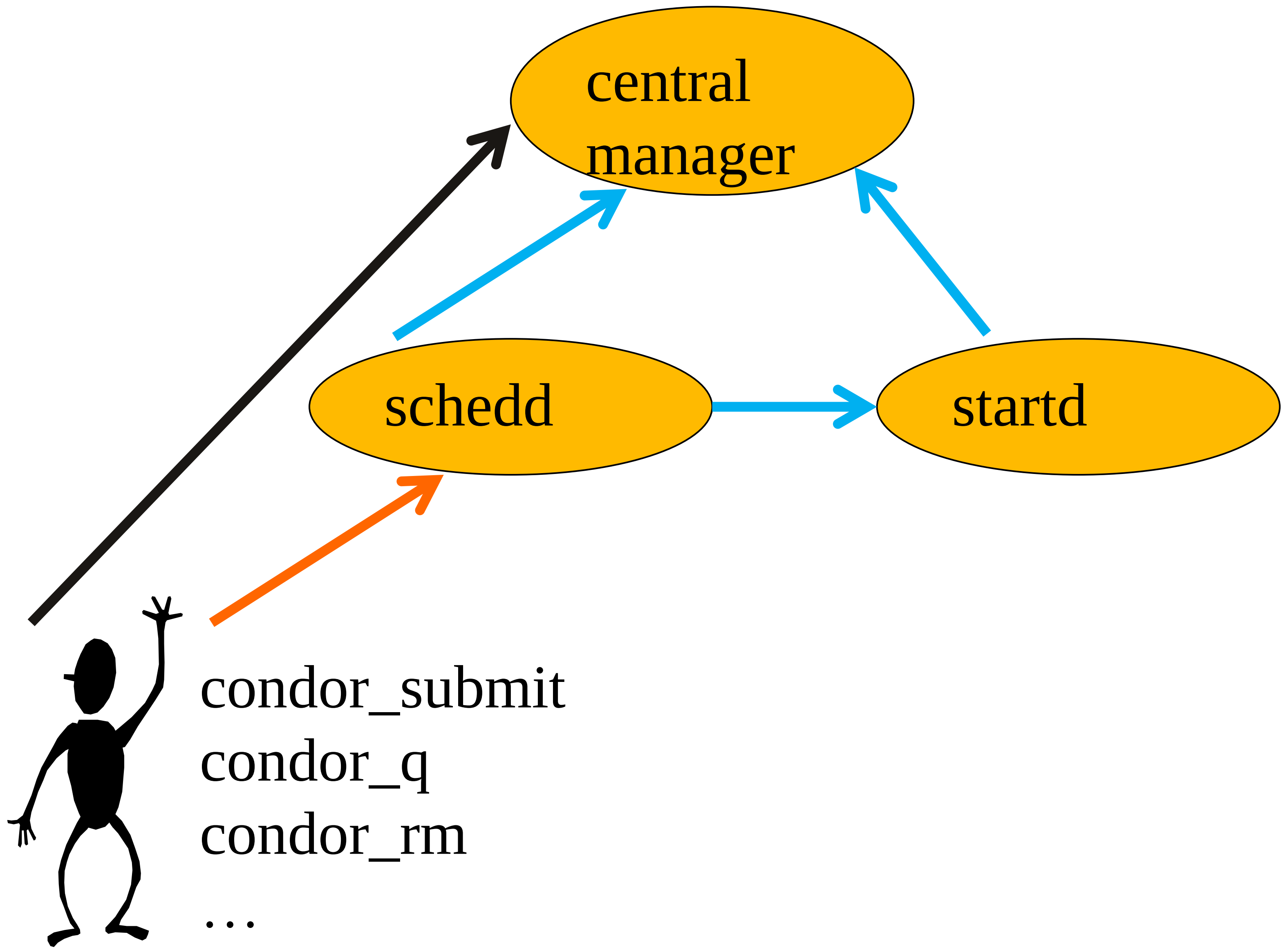
```
ALLOW_READ = *.wisc.edu
ALLOW_WRITE= *.wisc.edu
ALLOW_ADMINISTRATOR =
  admin@wisc.edu/*.wisc.edu,
  $(CONDOR_HOST)
```

Example Security Configuration

```
ALLOW_DAEMON =
  daemon@wisc.edu/*.wisc.edu
ALLOW_NEGOTIATOR =
  daemon@wisc.edu/$(CONDOR_HOST)
```

Users without Certificates

- Using **`FS`** authentication these users can submit jobs and view the queue on the local schedd
- **`condor_q -analyze`** and **`condor_status`** won’t work for normal users without an X.509 certificate
  - Requires **`READ`** access to **`condor_collector`**
- FS won’t work across the network!
- How to let anyone read any daemon?



condor_submit
condor_q
condor_rm
…

Allow Any User Read Access

- One option: Allow weak methods for READ:

```
SEC_READ_AUTHENTICATION_METHODS =
    FS, SSL, CLAIMTOBE
SEC_CLIENT_AUTHENTICATION_METHODS =
    FS, SSL, CLAIMTOBE
```

- Or, just don’t require authentication at all for READ commands:

```
SEC_READ_AUTHENTICATION = OPTIONAL
```

Example, on one page

```
SEC_DEFAULT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, SSL
SEC_DEFAULT_ENCRYPTION = REQUIRED
SEC_DEFAULT_INTEGRITY = REQUIRED
SEC_READ_AUTHENTICATION = OPTIONAL
SEC_DAEMON_AUTHENTICATION_METHODS = SSL

ALLOW_READ = *.wisc.edu
ALLOW_WRITE= *.wisc.edu
ALLOW_ADMINISTRATOR = admin@wisc.edu/*.wisc.edu, \
  $(CONDOR_HOST)
ALLOW_DAEMON = daemon@wisc.edu/*.wisc.edu
ALLOW_NEGOTIATOR = daemon@wisc.edu/$(CONDOR_HOST)
```

Todd's Shared Secret Formula

```
# Require authentication, encryption, integrity
use SECURITY: Strong

# By default, must authenticate via filesystem
# or pool password
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, PASSWORD

# Allow READ level access (e.g. condor_status)
# with ANONYMOUS authentication
SEC_READ_AUTHENTICATION_METHODS = \
   $(SEC_DEFAULT_AUTHENTICATION_METHODS), ANONYMOUS

# Have tools like condor_status attempt ANONYMOUS
# authentication so that condor_status will work
# from any machine in the pool.
SEC_CLIENT_AUTHENTICATION_METHODS = \
   $(SEC_DEFAULT_AUTHENTICATION_METHODS), ANONYMOUS
SEC_PASSWORD_FILE = /etc/condor/poolpassword
```

Conclusion

Attached to Indico is Zach's step-by-step securing via SSL with your own CA talk…

… but this is overly complex IMO. Plan on adding security cut-n-paste HOWTOs on wiki.htcondor.org… and hopefully some simpler 'meta-knobs' that lean more on convention than configuration.