Overview

› What are the threats?

› Who do you trust?

› What are the mechanisms?

› Other security concerns?
The purpose of HTCondor is to accept arbitrary code from users and run it on a large number of machines.
The purpose of HTCondor is to accept arbitrary code from users and run it on a large number of machines.

The purpose of a botnet is to take arbitrary code and run it on a large number of machines.
So what’s the difference?

You wish to prevent unauthorized access

Ultimately, it just comes down to who can use your pool, and how they can use it.
Basic Concepts

› “Who can use your pool” is really two concepts:

› The “Who” is authentication

› The “can use” is authorization
Basic Concepts

In the context of an HTCondor pool:

• You want only hosts (machines) that you trust to be in the pool
  • ^^^ Is that enough?

• You want only people you trust to submit jobs
Authentication

› For a secure pool, both users and HTCondor daemons must authenticate themselves

› HTCondor supports several mechanisms:
  • Host based (by just using source IP address)
  • File System (FS) – used by schedd by default
  • Pool Password (PASSWORD)
  • KERBEROS
  • SSL
  • GSI
Other Security Features

- In addition to authenticating network connections, you may also wish to use:
  - Integrity Checks (MD5)
    - Allows HTCondor to know if traffic has been tampered with
  - Encryption (3DES, Blowfish)
    - Allows HTCondor to transmit encrypted data so it cannot be spied on while in transit
Authorization Levels

› READ
› WRITE
  • submit jobs, ...
› DAEMON
  • Advertise ads into the collector, claim slots, ...
› ADMINISTRATOR
  • Change user priorities, reconfig
› NEGOTIATOR
  • Can give matches (slots) to schedds
When first contacting each other, HTCondor daemons have a short negotiation to find out which mechanisms are support and what features are required for the connection.
Security Negotiation

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
ENCRIPTION = YES
INTEGRITY = YES
AUTHENTICATION = YES
METHODS = SSL
CONDOR_HOST = my-central-manager.wisc.edu
ALLOW_READ = *

Central Manager

negotiator

collector

Submit Node

schedd

Worker Node

startd

job
CONDOR_HOST = my-central-manager.wisc.edu
ALLOW_READ = *
ALLOW_DAEMON = $(CONDOR_HOST), submit*.wisc.edu, worker*.wisc.edu
CONDOR_HOST = my-central-manager.wisc.edu
ALLOW_READ = *
ALLOW_DAEMON = $(CONDOR_HOST), submit*.wisc.edu, worker*.wisc.edu
ALLOW_NEGOTIATOR = $(CONDOR_HOST)
CONDOR_HOST = my-central-manager.wisc.edu
ALLOW_READ = *
ALLOW_DAEMON = $(CONDOR_HOST), submit*.wisc.edu, worker*.wisc.edu
ALLOW_NEGOTIATOR = $(CONDOR_HOST)
ALLOW_ADMINISTRATOR = $(CONDOR_HOST)
Thoughts?

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDOR_HOST</td>
<td>my-central-manager.wisc.edu</td>
</tr>
<tr>
<td>ALLOW_READ</td>
<td>*</td>
</tr>
<tr>
<td>ALLOW_DAEMON</td>
<td>$(CONDOR_HOST), submit*.wisc.edu, worker*.wisc.edu</td>
</tr>
<tr>
<td>ALLOW_NEGOTIATOR</td>
<td>$(CONDOR_HOST)</td>
</tr>
<tr>
<td>ALLOW_ADMINISTRATOR</td>
<td>$(CONDOR_HOST)</td>
</tr>
</tbody>
</table>
Create a pool password file and copy to all machines in the pool

Central Manager

negotiator → collector

Submit Node

schedd

Worker Node

startd

job
# require authentication and integrity for everything...
SEC_DEFAULT_AUTHENTICATION=REQUIRED
SEC_DEFAULT_INTEGRITY=REQUIRED
# ...except read access...
SEC_READ_AUTHENTICATION=OPTIONAL
SEC_READ_INTEGRITY=OPTIONAL

# this will require PASSWORD authentications for daemon-to-daemon, and
# allow FS authentication for submitting jobs and administrator commands
SEC_PASSWORD_FILE = /etc/condor/passwords.d/POOL
SEC_DEFAULT_AUTHENTICATION_METHODS = FS, PASSWORD
SEC_DAEMON_AUTHENTICATION_METHODS = PASSWORD
SEC_NEGOTIATOR_AUTHENTICATION_METHODS = PASSWORD

### AUTHORIZATION SECTION (eg ALLOW_*, DENY_*)
# allow any process that can read the pool password to act as a daemon
ALLOW_DAEMON = condor_pool@*
# allow admin commands from root or tannenba on the central manager
ALLOW_ADMINISTRATOR = root@*/$(CONDOR_HOST), \
tannenba@*/$(CONDOR_HOST)
# only the condor daemons on the central manager should be negotiating
ALLOW_NEGOTIATOR = condor_pool@*/$(CONDOR_HOST)
Pretty good... any "bad news"?
Could use Puppet SSL certs…

# Require SSL for daemon-to-daemon communications
SEC_DAEMON_INTEGRITY = REQUIRED
SEC_DAEMON_AUTHENTICATION = REQUIRED
SEC_DAEMON_AUTHENTICATION_METHODS = SSL
SEC_NEGOTIATOR_INTEGRITY = REQUIRED
SEC_NEGOTIATOR_AUTHENTICATION = REQUIRED
SEC_NEGOTIATOR_AUTHENTICATION_METHODS = SSL

# If you have a mapfile, set this to the HTCondor canonical name instead
ALLOW_DAEMON = ssl@unmapped

# SSL cert and key locations
SSL_DIR = /var/lib/puppet/ssl
AUTH_SSL_CLIENT_CAFILE = $(SSL_DIR)/certs/ca.pem
AUTH_SSL_CLIENT_CERTFILE = $(SSL_DIR)/certs/$(FULL_HOSTNAME).pem
AUTH_SSL_CLIENT_KEYFILE = $(SSL_DIR)/private_keys/$(FULL_HOSTNAME).pem
AUTH_SSL_SERVER_CAFILE = $(SSL_DIR)/certs/ca.pem
AUTH_SSL_SERVER_CERTFILE = $(SSL_DIR)/certs/$(FULL_HOSTNAME).pem
AUTH_SSL_SERVER_KEYFILE = $(SSL_DIR)/private_keys/$(FULL_HOSTNAME).pem

See security HOWTO Recipes at htcondor.org
Or could use new TOKEN authentication method

- New in HTCondor v8.9
- A token is signed by a symmetric private key (e.g. the pool password!) and contains
  - An identity
    - For use in ALLOW_XXX and DENY_XXX authorization lists
  - An expiration time
  - A bounding set of permitted actions
Keep pool password on CM...
Create some tokens...

condor_token_create -identity node1@pool.example.com \
  -lifetime 160000000 \n  -authz ADVERTISE_STARTD

condor_token_create -identity node2@pool.example.com \
  -lifetime 160000000 \n  -authz ADVERTISE_STARTD

condor_token_create -identity submit1@pool.example.com \
  -lifetime 160000000 \n  -authz ADVERTISE_SCHEDD

....
And distribute tokens....

Central Manager

negotiator

collector

Submit Node

schedd

Worker Node

startd

job
Or just use v8.9 security "Quick Configuration" for a new pool!

When installing a new pool, assuming you are on a trusted network and there are no unprivileged users logged in to the submit hosts:

1. Start HTCondor on your central manager host (containing the `condor_collector` daemon) first. For a fresh install, this will automatically generate a random key in the file specified by `SEC_PASSWORD_FILE` (defaulting to `/etc/condor/passwords.d/POOL` on Linux).

2. Install an auto-approval rule on the central manager using `condor_token_request_auto_approve`. This automatically approves any daemons starting on a specified network for a fixed period of time. For example, to auto-authorize any daemon on the network `192.168.0.0/24` for the next hour (3600 seconds), run the following command from the central manager:

   ```bash
   condor_token_request_auto_approve -netblock 192.168.0.0/24 -lifetime 3600
   ```

3. Within the auto-approval rule’s lifetime, start the `condor_schedd` and `condor_startd` hosts inside the appropriate network. The token requests for these daemons will be automatically approved and installed into `/etc/condor/tokens.d/`; this will authorize the daemon to advertise to the collector. By default, auto-generated tokens do not have an expiration.

This quick-configuration requires no configuration changes beyond the default settings. More complex cases, such as those where the network is not trusted, are covered in the `Token Authentication` section.
What about user submit from a different node? Or Jupyter NB?

Central Manager

- negotiator
- collector

Submit Node

- schedd

Worker Node

- startd

Condor_token_fetch -lifetime XXX
Configuration Security

› Are your condor_config files secured?

› They should be owned and only modifiable by root.

› If you use a config directory, make sure only root can create files in it
HTCondor can allow configuration changes using a command-line tool:

- `condor_config_val --set Name Value`

However, this behavior is off by default and needs to be enabled on a case-by-case basis for each config parameter… use carefully only if you really need it.
HTCondor Privilege

- HTCondor typically runs “as root”

- Why?
  - Impersonating users
  - Process isolation
  - Reading secure credentials

- When it isn’t actively using root, it switches effective UID to another user (“condor”)
HTCondor Privilege

› HTCondor will never launch a user job as root. There is a “circuit breaker” at the lowest level to prevent it.

› If not using system credentials, the Central Manager can run without root priv

› Let’s examine some different Startd configurations
StartD Configurations

Startds have a few different options for running jobs by comparing UID_DOMAIN:

› Run jobs as the submitting user

› Run jobs as a dedicated user per slot
  • Keeps jobs running as a low-privilege user
  • Isolates jobs from one another
  • Makes it easy to clean up after a job

› Run jobs as the user “nobody”
  • May allow jobs to interfere with one another
  • This helps: USE_PID_NAMESPACES = True
Even if that admin has not required encryption for all network connections, user jobs can specify per-file for both input and output if the files should be encrypted:

- `Encrypt_Input_Files = file1, *.dat`
- `Encrypt_Output_Files = data.private`
Encrypt Execute Directory

› If you are using Linux with *ecryptfs* installed, you can have HTCondor encrypt the execute directory on disk, offering extra protection of sensitive data.

› Can be enabled pool-wide by the admin:
  • ENCRYPT_EXECUTE_DIRECTORY = True

› Per-job in the submit file:
  • Encrypt_Execute_Directory = True
Restricting Users

 › SUBMIT_REQUIREMENT allows the administrator to restrict what jobs are able to enter the queue

 › Can be used to prevent users from lying about what groups they belong to:

 ```
 SUBMIT_REQUIREMENT_NAMES = GROUP1

 SUBMIT_REQUIREMENT_GROUP1 = (AcctGroup =!= "group1") ||
 (AcctGroup =?= "group1" && (Owner=="zmiller" || Owner=="tannenba"))

 SUBMIT_REQUIREMENT_GROUP1_REASON = "User not in group1"
 ```
SUBMIT_REQUIREMENT allows the administrator to restrict what jobs are able to enter the queue.

Can be used to allow only certain executable files, number of CPUs requested for a job, anything else that is part of the Job ClassAd.
Vulnerabilities

› HTCondor is periodically assessed by an independent research group.

› Our vulnerability reporting process is documented and vulnerability reports publicly available:
  • http://research.cs.wisc.edu/htcondor/security/
Thank you and Questions?

› Tip: Try emailing the htcondor-users mailing list:

https://lists.cs.wisc.edu/mailman/listinfo/htcondor-users