“Forget what you know about HTCondor security. We changed it”
Establishing a secure pool

Traditionally, there’s been no “easy button” to setup strong security on pools.

- Very easy to setup poor security.
- Very hard to find good advice on strong security.
- No tools provided by HTCondor to setup strong authentication.
  - Is the answer “Google for how to create a new CA with OpenSSL”?

In 2019, we spent significant blood, sweat, and tears providing a new authentication method and new tooling to setup your pool.
Classic HTCondor Daemon Security

For invoking a remote command:

- The server and client would negotiate an authentication method to establish identities.
  - Example methods: GSI, PASSWORD, SSL.
- Once an identity was established, HTCondor would determine if the requested command was authorized.
  - Can user foo@example.com perform actions that require DAEMON-level authorization?!

Each pair of daemons that want to talk had to perform this dance!
How do we setup SSL security for HTCondor?
Setting up SSL Security

Step 0. Figure out this is a thing you want to do!

... oodles of old presentations to go through. Which one is right?
Setting up SSL Security

1a. Find Zach Miller’s HTCondor Week 2011 talk.

1b. Whoops
Setting up SSL Security

1. Ask Google for Help
2. Distribute CA across the cluster

```bash
# SSL settings
AUTH_SSL_SERVER_CERTFILE=/etc/grid-security/condor/hostcert.pem
AUTH_SSL_SERVER_KEYFILE=/etc/grid-security/condor/hostkey.pem
AUTH_SSL_SERVER_CADIR=/etc/grid-security/certificates
AUTH_SSL_CLIENT_CADIR=/etc/grid-security/certificates

# Authentication settings
SEC_DEFAULT_AUTHENTICATION = REQUIRED
SEC_READ_AUTHENTICATION = OPTIONAL
SEC_CLIENT_AUTHENTICATION = OPTIONAL
SEC_DEFAULT_AUTHENTICATION_METHODS = GSI
SCHEDD_SEC_WRITE_AUTHENTICATION_METHODS = FS,GSI
SCHEDD_SEC_DAEMON_AUTHENTICATION_METHODS = FS,GSI
SEC_CLIENT_AUTHENTICATION_METHODS = FS,GSI

# Privacy settings
SEC_DEFAULT_ENCRYPTION = OPTIONAL
SEC_DEFAULT_INTEGRITY = REQUIRED
SEC_READ_INTEGRITY = OPTIONAL
SEC_CLIENT_INTEGRITY = OPTIONAL
SEC_READ_ENCRYPTION = OPTIONAL
SEC_CLIENT_ENCRYPTION = OPTIONAL
```

... and add some modest configs
... and distribute host certificates everywhere
Setting up SSL Security

2. Distribute CA across the cluster
3. Configure schedd -> collector auth
4. startd -> collector auth
5. negotiator -> schedd
6. schedd -> startd

# Mapfile
CERTIFICATE_MAPFILE = /etc/condor/condor_mapfile

# Authorization settings
# These should be unnecessary, unless if we have an error below.
DENY_WRITE = anonymous*, unmapped*
DENY_NEGOTIATOR = anonymous*, unmapped*
DENY_ADMINISTRATOR = anonymous*, unmapped*
DENY_DAEMON = anonymous*, unmapped*

# Macros defining group of daemons
FRIENDLY_DAEMONS = *daemon.example.com
WORKER_NODES = *worker.example.com/
USERS = *@example.com

# Authz settings for each daemon. Preferably, change the templates above
DEFAULT_WRITE = $(FRIENDLY_DAEMONS), $(HOSTNAME)@worker.example.com/$(FULL_HOSTNAME)
ALLOW_WRITE = $(DEFAULT_WRITE)

# Schedd is the only one accepting non-strong auth
SCHEDD_ALLOW_WRITE = $(USERS), $(HOSTNAME)@daemon.example.com/$(FULL_HOSTNAME)
NEGOTIATOR_ALLOW_WRITE = $(FRIENDLY_DAEMONS)
COLLECTOR_ALLOW_ADVERTISE_MASTER = $(FRIENDLY_DAEMONS), $(WORKER_NODES), condor@example.com
COLLECTOR_ALLOW_ADVERTISE_SCHEDD = $(FRIENDLY_DAEMONS)
COLLECTOR_ALLOW_ADVERTISE_STARTD = $(WORKER_NODES), $(HOSTNAME)@daemon.example.com/$(FULL_HOSTNAME)
STARTD_ALLOW_NEGOTIATOR = cluster-condor@daemon.example.com/cluster-condor.example.com
SHADOW_ALLOW_WRITE = $(DEFAULT_WRITE), $(WORKER_NODES), $(HOSTNAME)@daemon.example.com/$(FULL_HOSTNAME)
ALLOW_DAEMON
ALLOW_ADMINISTRATOR

7. ... Or just give up and use host security?
For a few years, HTCondor has had “match password” security. In this case,

- The startd generates a capability, $T$, and sends it to the collector in its ClassAd.
  - Anyone with $T$ is allowed to start jobs on the startd.
  - The negotiator gets $T$ from the collector because the collector trusts the negotiator.
  - The schedd gets $T$ as part of the ‘match’ created by the negotiator.
  - Hence the name “match password”.

The startd trusts any negotiator trusted by its collector!

The negotiator trusts any schedd in the collector.

The startd trusts any schedd in its collector.
Extending Trust in the Collector

Starting in 8.9.x, the schedd also generates a token, T”, and sends it in its ClassAd.

• The schedd trusts the collector only gives T” to trustworthy negotiators.

• Any client with T” is allowed to be a negotiator for the schedd.

Note the only “full authentication” arrows are to the collector.
In 8.9, all trust is established is through the collector.

- Instead of needing credentials between any two daemons, only credentials to authenticate with the collector are needed.
  - We implicitly trust anyone the collector hands our security sessions to.
  - Think of the collector as establishing a trust domain.
  - Trust domain -> set of daemons run by the same administrator.

If we trust the collector, why not allow it to issue credentials?
Introducing:

The IDTOKEN

eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.eyJpYXQiOjE1ODk1NjYwOTEsImlzcyI6ImNvbGxlY3Rvci5leGFtcGxlLmNvbSIsImp0aSI6ImQyODI1YjNhYTkyNzcyYWQ3ZmJiNmNmMDNmZmI0ZmU2Iiwic3ViIjoiYnJpYW4uYm9ja2VsbWFuQGNvbGxlY3Rvci5leGFtcGxlLmNvbSJ9.z8LUtjmqL_bqXTtUpC0-nXGflBfW3zI0JuB43S9MOGE

Many of the ideas were extensions of approaches developed as part of SciTokens (NSF #1738962)
An IDTOKEN is a bearer token that can be used to authenticate an identity:

- An **IDTOKEN is signed** (often by the collector) – the signature can be validated by a daemon with the master password.
- Any given token is valid within a single trust domain.
  - Multiple master passwords can be used within the same trust domain.
- The **IDTOKEN embeds an identity**. The HTCondor authorization system can operate on this identity.
- The IDTOKEN may contain restrictions:
  - On when the token is valid (“expires next week”)
  - What the token can be used for (“useful only for READ permission”).

Any client can have multiple IDTOKENS – useful for authenticating with servers in different trust domains!
Big secret: IDTOKENS are JWTs - Basic Example

The trust domain ("iss" -> issuer)

When the token was issued

Unique ID

User identity

(output from jwt.io)
The IDTOKEN contains an identity within HTCondor.

There’s no “mapfile” as in SSL/GSI as there’s no external identity to map.

You do have to authorize an identity to perform an action (ALLOW_* options)
Big secret: IDTOKENS are JWTs – Complex Example

Encoded
PASTE A TOKEN HERE

Decoded
EDIT THE PAYLOAD AND SECRET

```
eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.e
yJleHAi0JEl0Tk4Mjc3NzUsIm1hdCI6MTU4OTgyND
E3NSwiaXNzIjoiY29sbGVjdG9yLmV4YW1wbGUuY
29tIiwianRpIjoiODkzNGYxZjQ5OWRiYzRmYzM0Z
TI1NmI1NzdOWUyN2MiLCJzdY29wZSI6ImNvbmbRvc
jovUkBRCIsInN1YiI6ImJyaWFuLmJvY2tlbgIhb
kBj2xszWN0b3luZXhhbXBsZS5jb20iKy
```

```json
{
  "alg": "HS256",
  "kid": "POOL"
}
```

Payload: DATA

```json
{
  "exp": 1589827775,
  "iat": 1589824175,
  "iss": "collector.example.com",
  "jti": "8934f1f499dbc4fc34e256b577a9e27c",
  "scope": "condor:/READ",
  "sub": "brian.bockelman@collector.example.com"
}
```

Limit on lifetime ("exp" = expiration)

Limit on authz

(output from jwt.io)
To see trust domains, tokens, and passwords in action, consider the case of schedd flocking:

- Each pool is a different trust domain – the two pool administrators are distinct!
- Accordingly, each collector has a separate master password.
- Each token is in one trust domain (and signed by a different password) so the schedd needs two tokens – one for each pool!
Where to find my token information

Each token is a file in a directory:
- ~/.condor/tokens.d/ (users)
- /etc/condor/tokens.d/ (condor or root)
- Overridden by SEC_TOKEN_DIRECTORY

Each master password is also a file in the directory:
- /etc/condor/passwords.d/
- Overridden by SEC_PASSWORD_DIRECTORY

The trust domain is configured by
- TRUST_DOMAIN
- Defaults to $(COLLECTOR_HOST)
Bootstrapping Trust – Creating an IDTOKEN

Anyone who can read the master password can issue any token they want using `condor_token_create`.

```
$ sudo condor_token_create \
  -identity brian.bockelman@collector.example.com \ 
  -lifetime 3600 \ 
  -authz READ -authz WRITE
```

eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.eyJleHAiOjE1ODk4Mjk4MzUsImIiOiJmIiwiYiI6IjYwMjE1MDc3NjI2Iiwic2NvcGUiOiJjb25kb3I6L1JFQUQgY29uZG9yOi9XUklURSiLuN1Yi6ImJyaWFuLmJvY2tlbG1hbkBjb25keS50cnVzdC50cyIiLCJpYiI6ImZyZWRpZS50cnVzdC50cyJ9
`
Fetching an IDTOKEN

Does authentication work now – but you need to squirrel away an IDTOKEN for future use? **condor_token_fetch** to the rescue!

- This tool authenticates with a daemon and asks the daemon to sign a token on behalf of the user’s identity. Resulting identity is identical to authenticated ID.
- **Use case:** I have an SSH login to a local schedd but want to remotely submit to a schedd in the same trust domain.
Want to get an IDTOKEN on a machine without authenticating?

- **condor_token_request** allows an anonymous user to request a token for an arbitrary identity $X$.
  - The token request can be approved either by an admin or a user authenticated as $X$.
  - Anyone can ask. Few can approve!
- **Use case:** I have an SSH login on a schedd and want to start submitting jobs from my laptop.
  - **Solution:** Request a token from my laptop; login to the submit host and approve the request.
- **DO NOT COPY/PASTE TOKENS.** Instead, use `condor_token_request`!

The startd, master, and schedd will automatically request tokens from the collector if authentication fails.
Bootstrapping Trust – Autoapproval

Token requests are a handy tool for securely bootstrapping authentication – provides a tool for secure movement of credentials.

• Great tool for adding a new worker node by hand.
• Crappy tool for adding 1,000 worker nodes!

Auto-approval mode for the rescue! Automatically approves certain requests for a specific network and time duration

How do I install a new cluster?

• On first start, a collector automatically generates a new master password.
• Enable auto-approval mode for hosts coming from the new subnet.
• If a schedd, startd, or master cannot authenticate with the collector, they will automatically request a token. If the requests come from the correct subnet, the token will be immediately issued.
• Wait for all the hosts to show up in condor_status then disable auto-approval mode.
Token Revocation

How do you “undo” a token? What happens if a user says “sorry, but my laptop got stolen and my IDTOKEN was on it?”

• An IDTOKEN must be signed by a password.
  • Remove the password and the token is invalid!
  • Erm … but all tokens are invalid.

• You can add a user to the DENY_* lists to remove all authorizations for that identity.
  • But then the user can never use HTCondor again 😞

• In 8.9.7, HTCondor has SEC_TOKEN_BLACKLIST_EXPR.
  • Any token matching this ClassAd expression will be rejected.
  • If you know the token unique ID, you can blacklist only that token. 😊
  • You can also reject all tokens owned by a specific user that were issued before last Friday. 😊
IDTOKENS strives to balance simplicity and fine-grained auth.

- Importantly, we provide tools to setup your HTCondor pool with IDTOKENS.

Not all is unicorns and rainbows!

- Some workflows still require central manager -> worker node auth: think `condor_fetchlog` or `condor_defrag`.
  - Giving an IDTOKEN to the worker node doesn’t help because IDTOKENS identify clients; here, the worker node is a server and needs the master password.
  - Should be fixed before 9.0.

- Documentation exists in the manual but it could admittedly use some love.

Would love to hear how you use these new tools!