Security and Token Auth Debugging

Brian Bockelman,

HTC23, Madison, WI

PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

iris hep

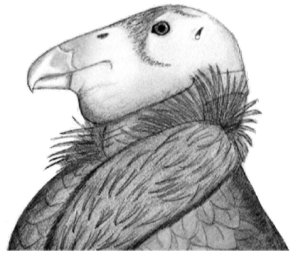
OSG

CHTC Center For High Throughput Computing

Goals for today:

- Learn the fundamental ‘jargon’ associated with access control in HTCSS.
- Understand the parts of the authorization handshake between client and server.
- Get a few debugging tips specific to token authentication.
- Learn the how security credentials are mapped into HTCondor identifiers and then authorized

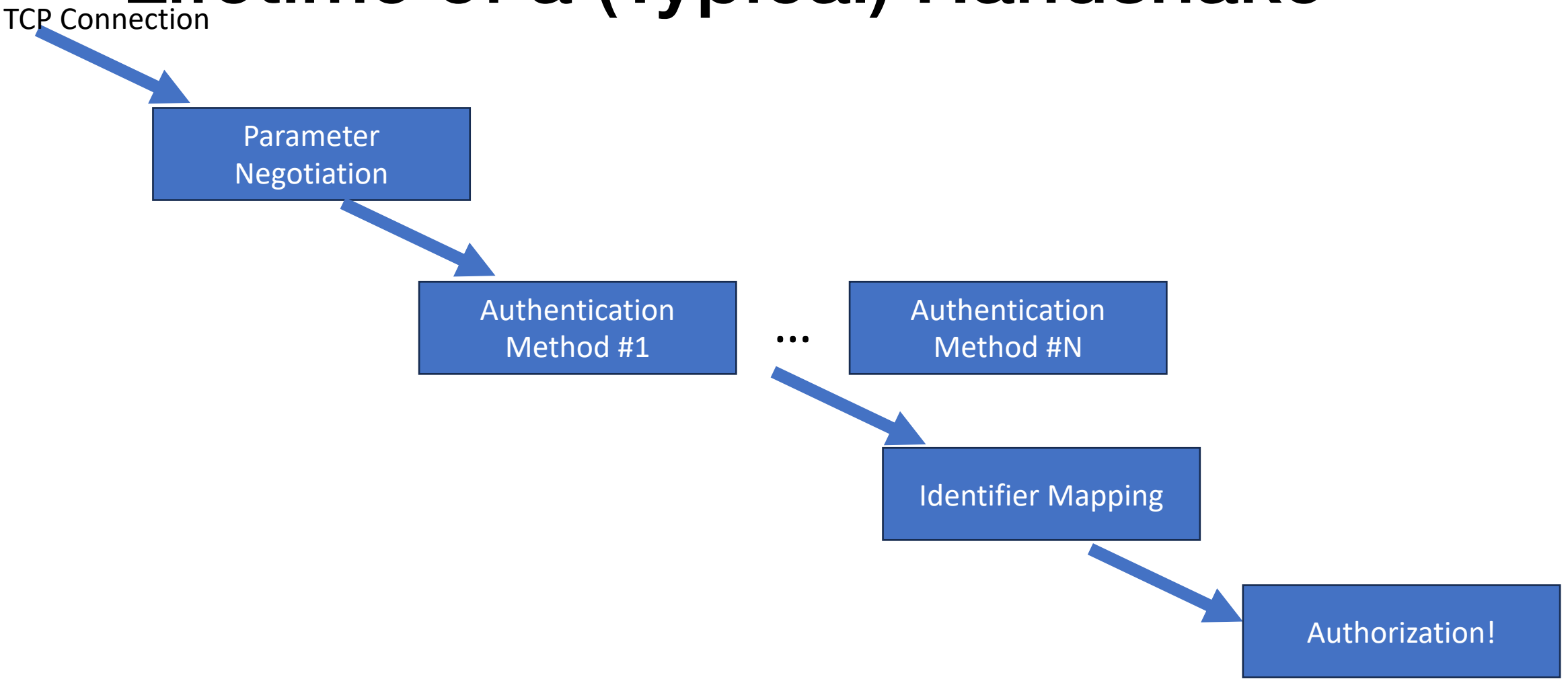
The Basic Vocabulary



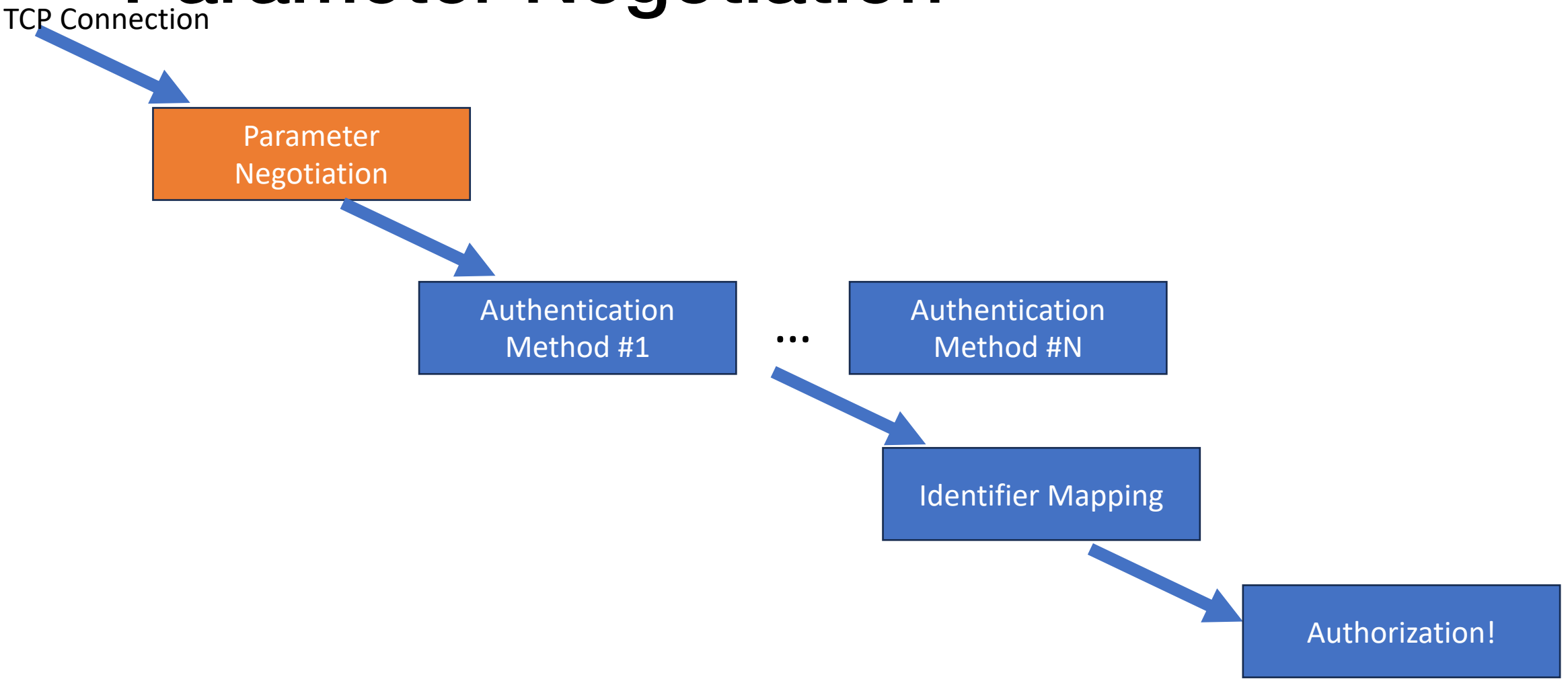
HTCondor Software Suite

- Authentication: Establishing an ‘identifier’ for a remote entity.
- Identity Mapping: Mapping between identifiers, such as from a Kerberos credential to a HTCondor identifier.
- Authorization: Determining whether an entity is permitted to perform a certain operation.
- Encryption: Maintaining confidentiality during a session.
- Integrity: Detecting modifications to a session done during transit.

Lifetime of a (Typical) Handshake

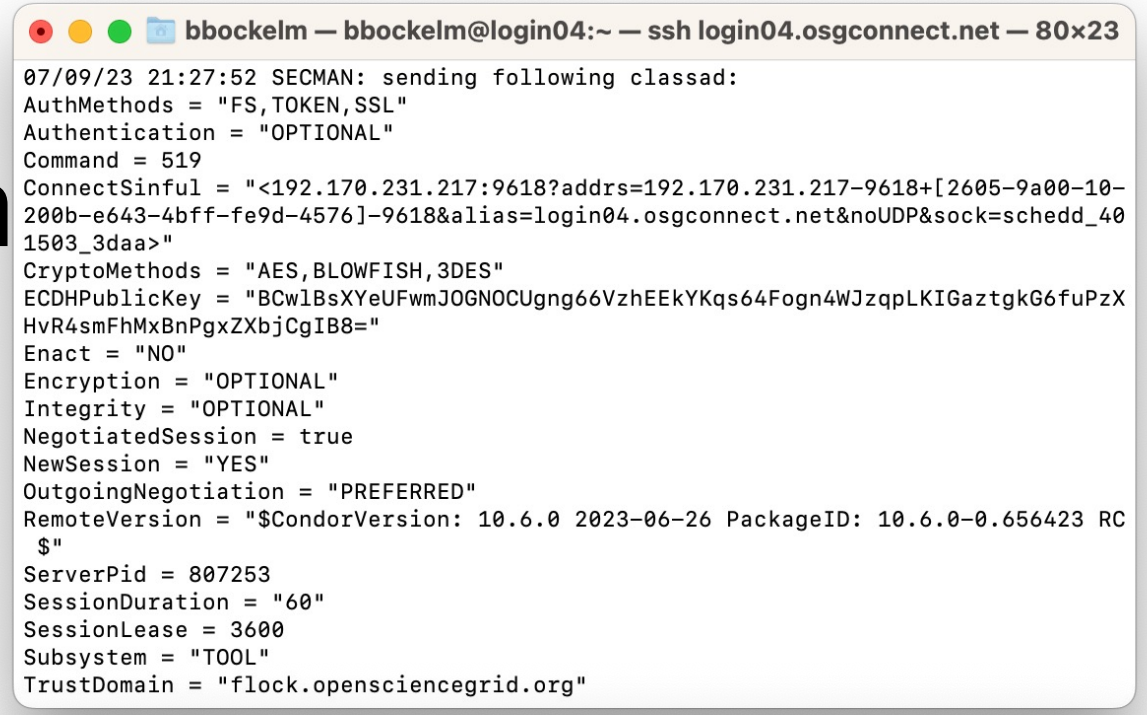


Parameter Negotiation

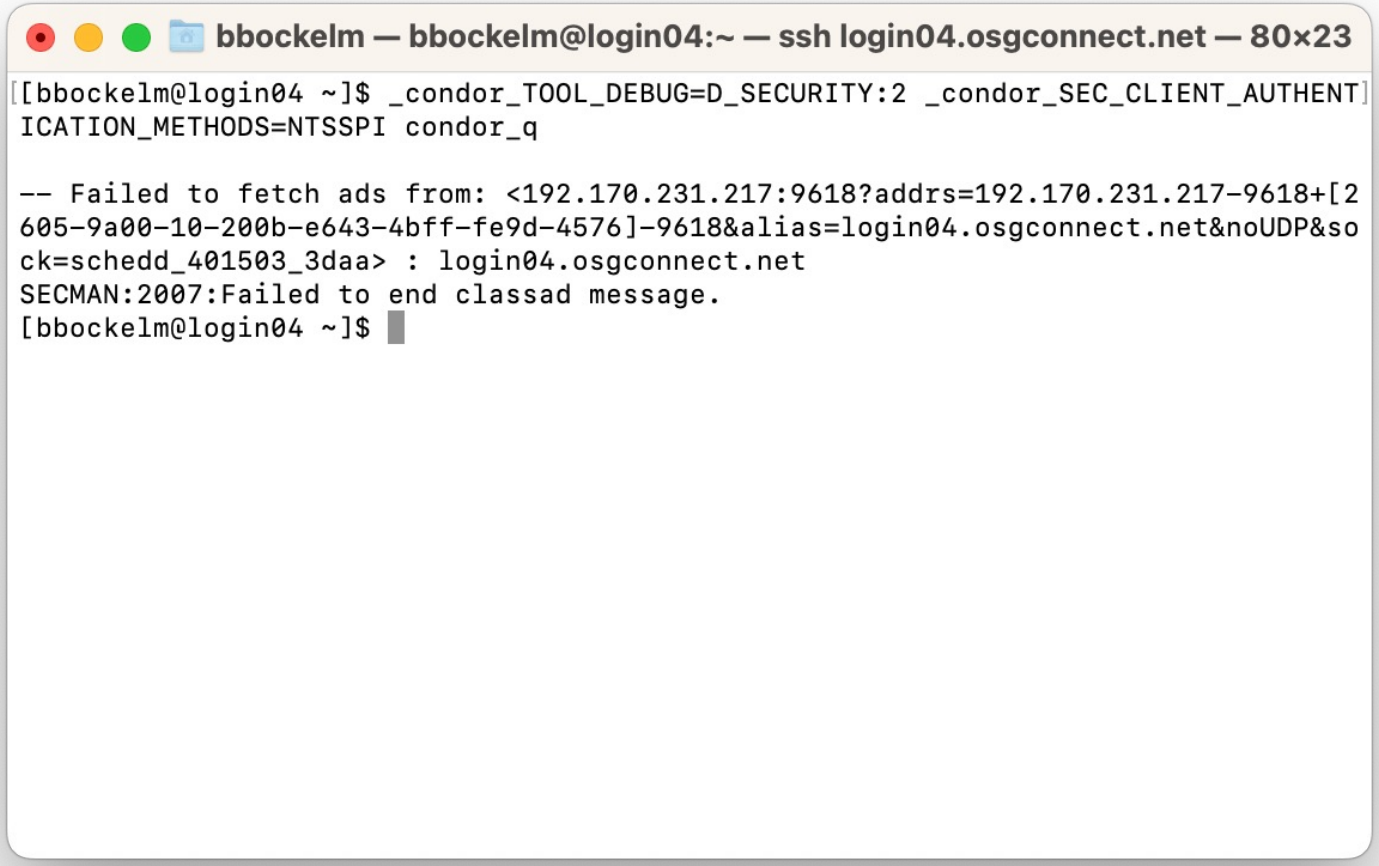


Parameter Negotiation

- Parameter negotiation consists of a ClassAd sent from client to server followed by one from server to client.
- Each side states their policy on topics like:
  - Is authentication required?
  - What authentication methods should be used?
  - Should encryption / integrity checking be used?



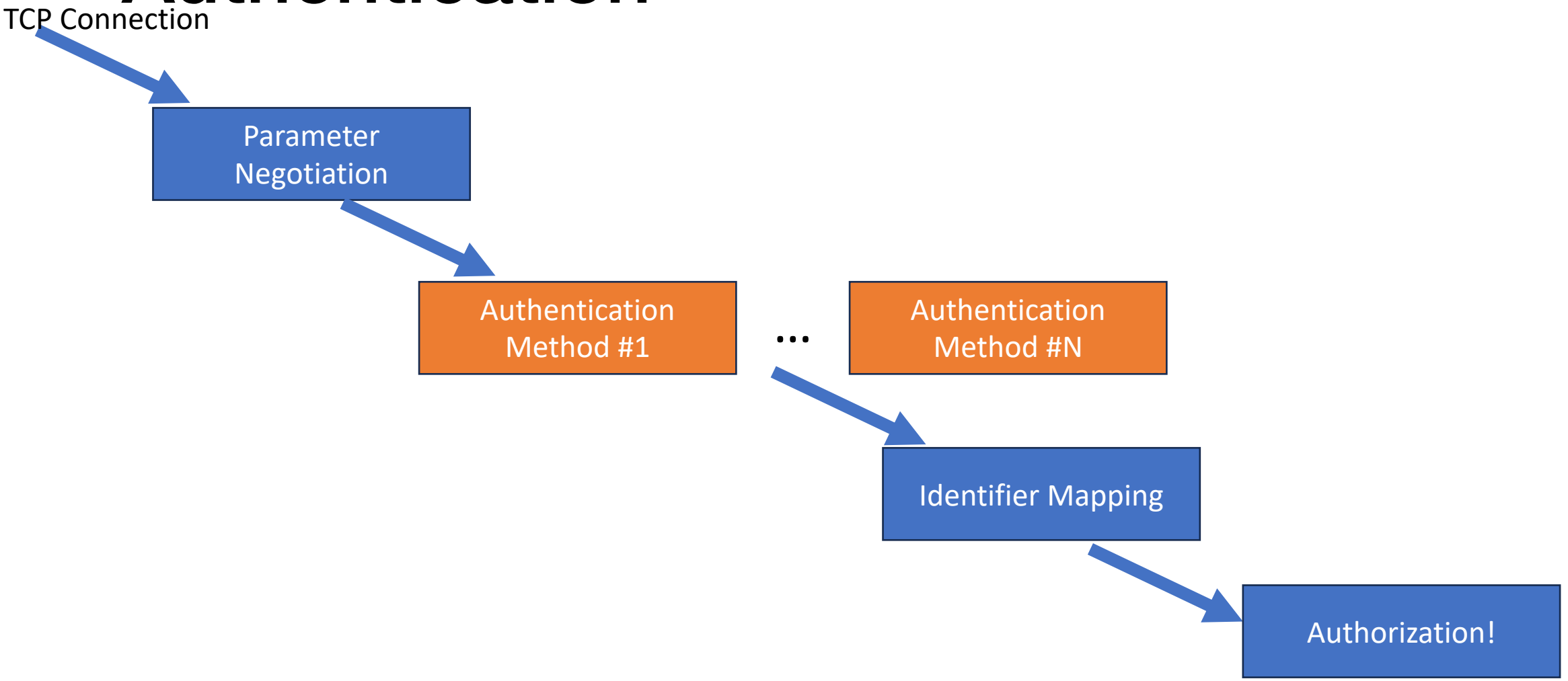
Parameter Negotiation



- It is possible for the client and server to have incompatible policy (example: no common methods).
- In this case, the server will abruptly close the socket. The client will report the dreaded

```
SECMAN:2007:Failed to end
classad message
```

Authentication



Authentication Methods

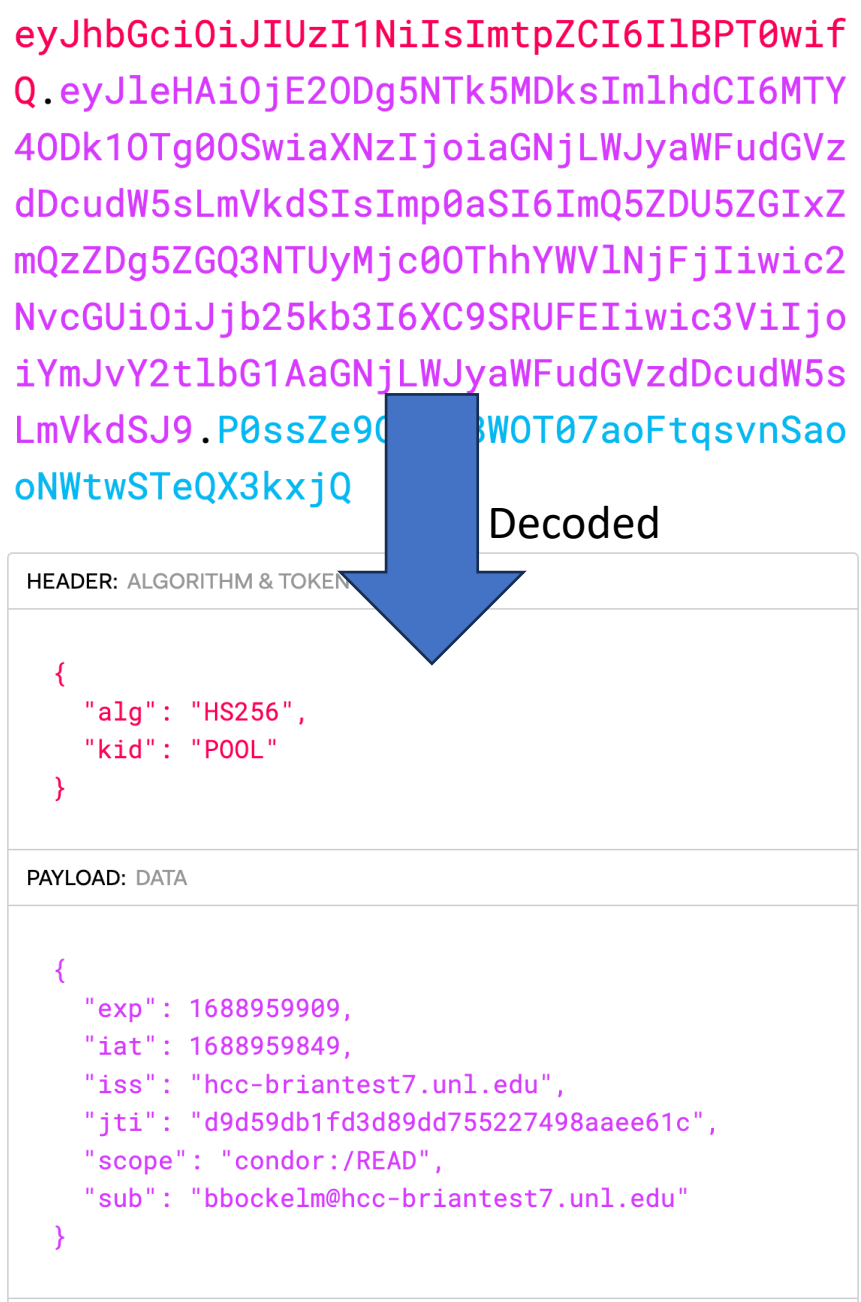
Will be covered today

- IDTOKENS: Client authenticates with a JSON Web Token (JWT) signed by the server.
- SCITOKENS: Client authenticates with a SciToken (https://scitokens.org/) JWT, signed by a trust third party.
- SSL: Client and server uses the venerable TLS protocol, same as HTTPS.
- KERBEROS: Client and server use Kerberos authentication.
- And other, less-commonly-used options:
  - NTSSPI, PASSWORD, CLAIMTOBE, ANONYMOUS, FS_REMOTE

Authentication: IDTOKENS

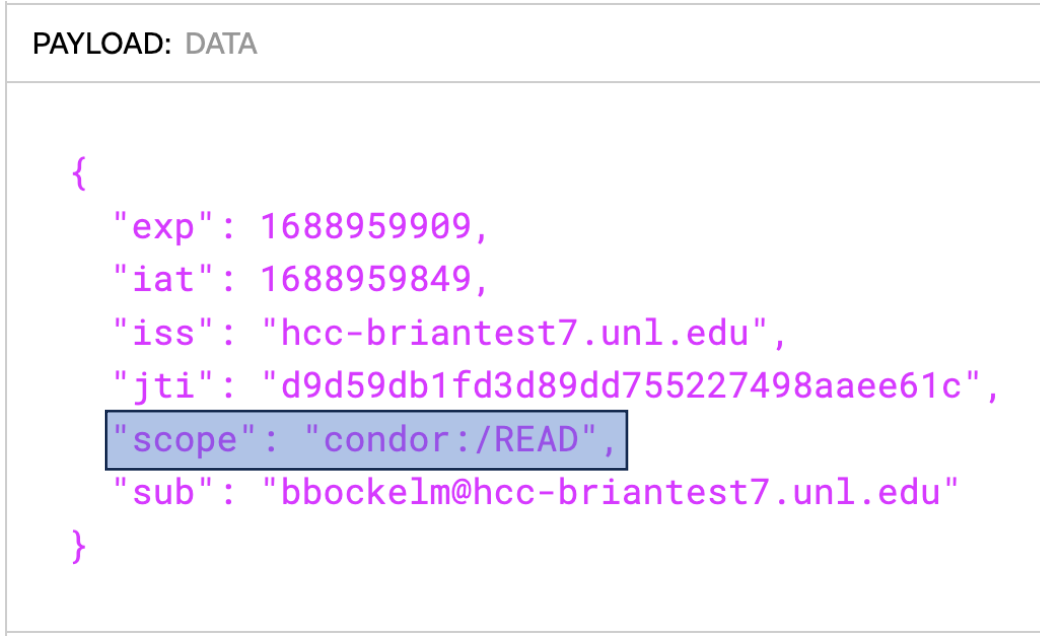
Authentication Protocol

- The IDTOKEN is used to establish a shared secret. The public portion is sent to the server; if the server has the right key, then it can regenerate the signature.
- Now, both sides have a shared secret (the token signature) and can use a key exchange protocol (AKEP2) to demonstrate possession to the other side.
- The client is identified by the subject in the token.



IDTOKENS - Authorizations

- As the IDTOKEN is generated by HTCondor, the ‘subject’ in the token is considered an HTCondor identifier
  -> No mapping step!
- Tokens *can* contain restrictions on allowed authorizations.
  - Can only act as a *restriction* on what the token can otherwise do – does not *grant* access beyond what’s configured.

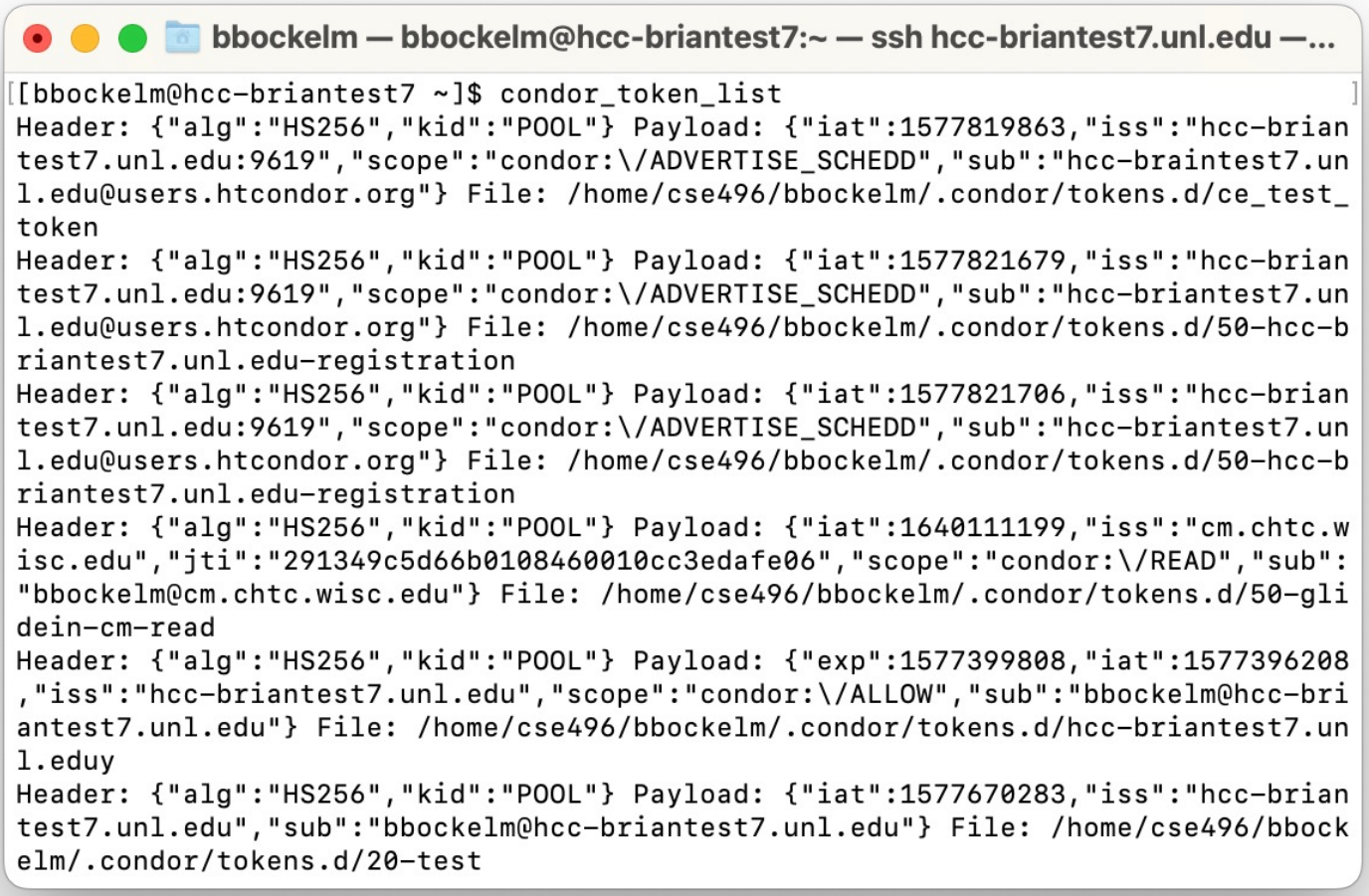


Key Concept – Trust Domains

- If the IDTOKEN subject is “native” to HTCondor, which instance?
  - After all, my identifier (“bbockelm”) for CHTC is different than for the OSPool (“brian.bockelman.1”)!
- We have added the concept of “Trust Domain” – the set of all services that are run by the same administrators.
  - We assume a named signing key in a trust domain always has the same value.
- Each server belongs to a given trust domain. It’ll ignore tokens from a different trust domain.

Trip Hazard: In 10.0.0, the default value of TRUST_DOMAIN changed! Check your tokens are still valid if you started using tokens in 9.x.

Finding the IDTOKEN



- There's a defined directory that holds tokens for a client (typically, `~/.condor/tokens.d`). The client iterates through each token in the directory, using the first one that "matches":
  - "Matches" means the token is in the same trust domain as the server and signed with a key the server knows.

Authentication: SciTokens

Authentication Protocol

- A TLS connection is established between client and server.
- The client verifies the server's host certificate.
- The client sends the SciToken across the secure channel.
- The server verifies the token was signed using the issuer's public key.
- Note:
  - The client authenticates the server via TLS.
  - The server authenticates the client using the token.

Finding the SciToken

- The client will send the token it finds in its environment using the Bearer Token Discovery protocol.
- Short version:
  - Look at the contents of the $BEARER_TOKEN environment variable.
  - Look at the contents of the file referred to by $BEARER_TOKEN_FILE.
  - Look at the contents of $XDG_RUNTIME/bt_u$UID
  - Look at the contents of /tmp/bt_u$UID
- The first token discovered is used; no matching is performed as in IDTOKENS.

Compare and Contrast: Token Auth'n

IDTOKENS-Specific

- Signed by the server (or whoever holds the symmetric key).
- Only verified by the same symmetric key.
- Discovery in a well-known user directory.
- Not sent to server; used to establish a shared secret.

Common

- Token format is JWT; can be introspected with any common JWT tools.
- Token contains common JWT attributes: expiration, validity time, subject/identifier.

SCITOKENS-Specific

- Signed by third-party JWT issuer.
- Verified by anyone who can download the public key.
- Discovered via WLCG Bearer token discovery protocol.
- Sent to server over TLS (Server needs host certificate)

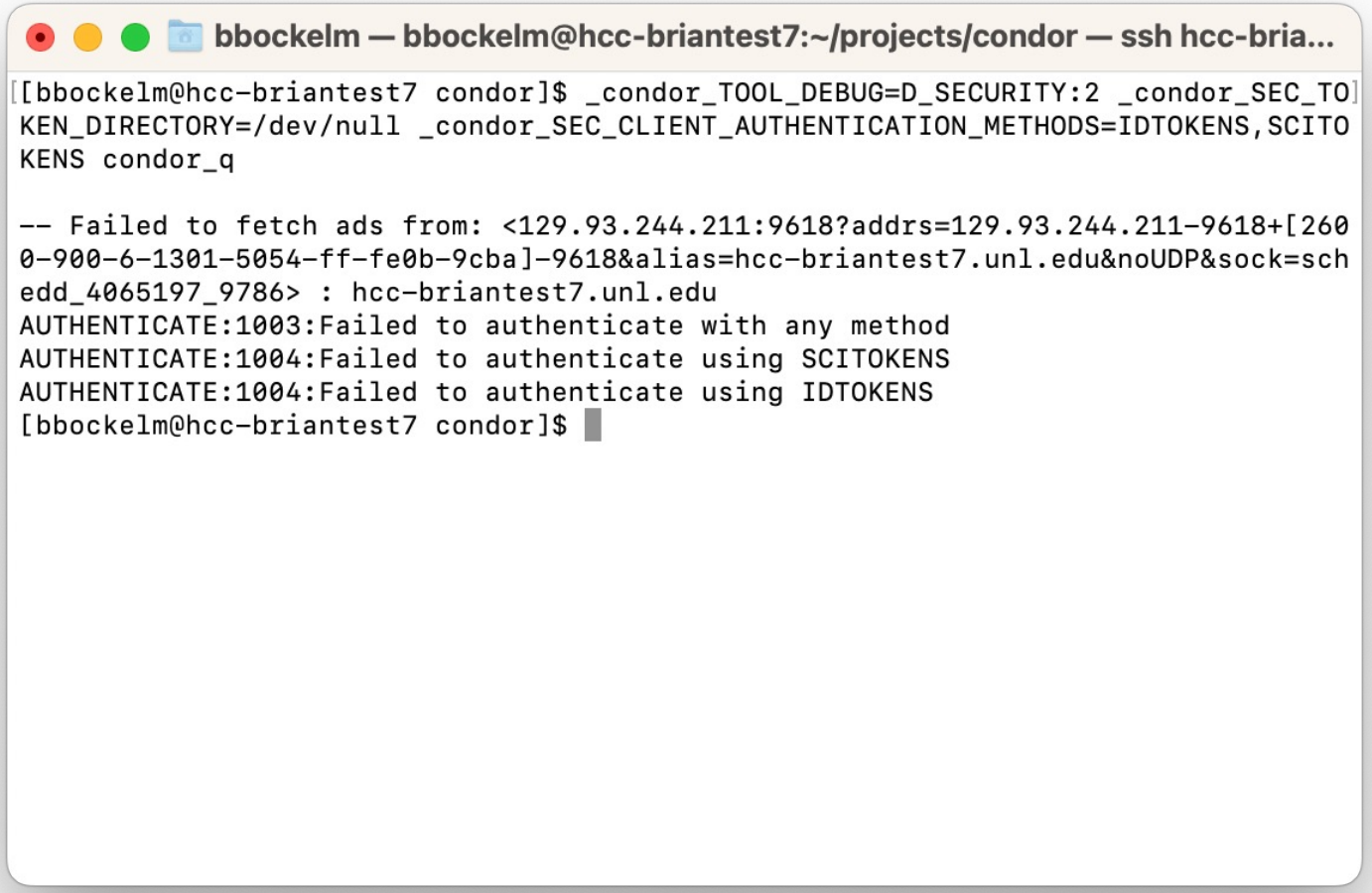
Authentication: SSL

Authentication Protocol

- Well … you know … TLS!
  - It is framed using HTCSS's CEDAR protocol, not raw TCP sockets. Cannot debug this with "openssl s_client".
  - About every 2 years we review the TLS crypto parameters to ensure they are modern (e.g., no MD5!).
- Client certificate / RFC 3820 proxy certificate is optional; the server can be configured to require one, however.
- Client certificate is discovered only using the value of AUTH_SSL_CLIENT_CERTFILE/AUTH_SSL_CLIENT_KEYFILE; does not follow Globus conventions

Authentication Failures

- When authentication fails, the client tool prints out every method it tried.
- It does not print out the failure reason for any of the protocols.
  - Sometimes this is because the server provides no error message about the rejection.
- Enabling security debug logging (D_SECURITY:2) is necessary to debug authentication failures.



Authentication Failures

bbockelm — bbockelm@hcc-briantest7:~ — ssh hcc-briantest7.unl.edu —...

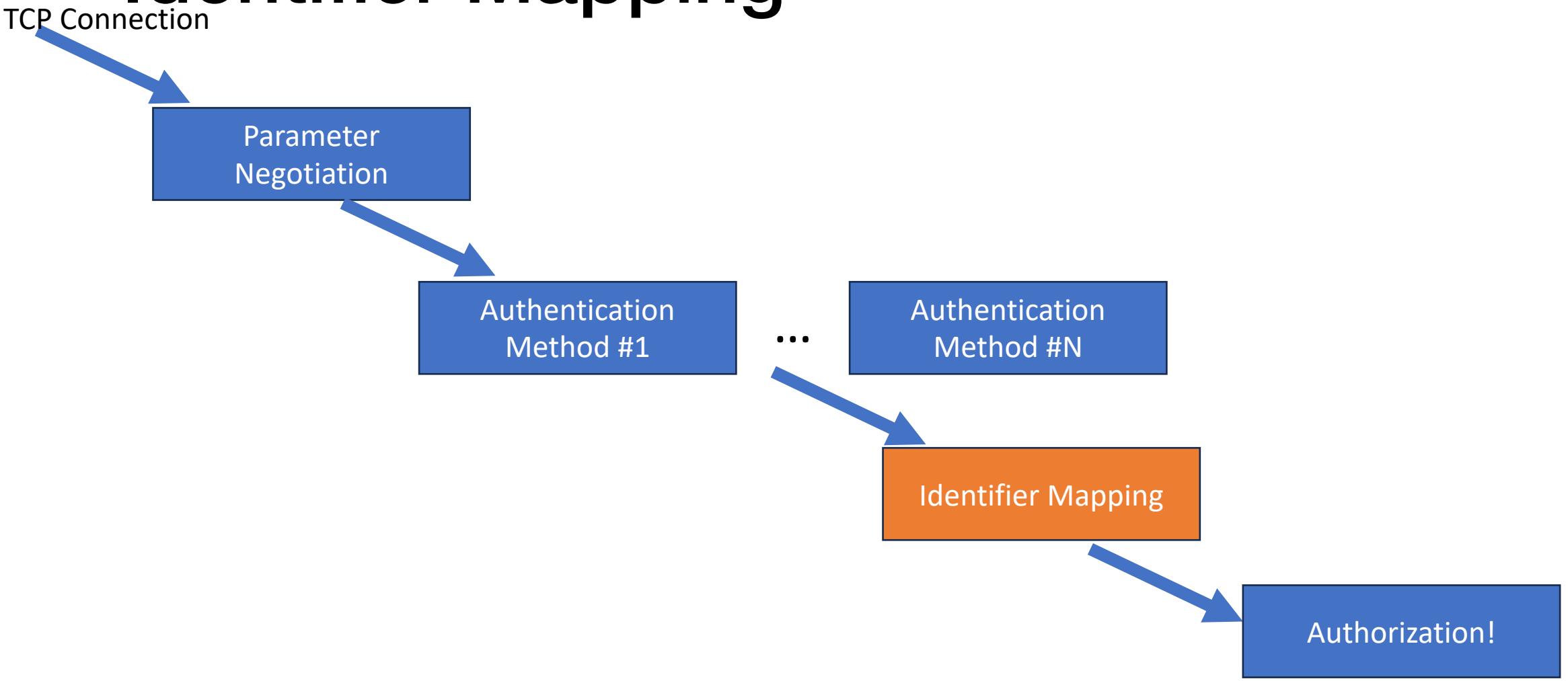
```
[bbockelm@hcc-briantest7 ~]$ _condor_TOOL_DEBUG=D_SECURITY:2 _condor_SEC_CLIENT_
AUTHENTICATION_METHODS=SSL condor_q

-- Failed to fetch ads from: <129.93.244.211:9618?addrs=129.93.244.211-9618+[260
0-900-6-1301-5054-ff-fe0b-9cba]-9618&alias=hcc-briantest7.unl.edu&noUDP&sock=sch
edd_1778306_94d1> : hcc-briantest7.unl.edu
AUTHENTICATE:1003:Failed to authenticate with any method
AUTHENTICATE:1004:Failed to authenticate using SSL
[bbockelm@hcc-briantest7 ~]$
```

bbockelm — bbockelm@hcc-briantest7:~ — ssh hcc-briantest7.unl.edu —...

```
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) SSL Auth: Trying to connect.
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) -Error with certificate at dept
h: 0
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY)   issuer   = /C=US/O=Let's Encr
ypt/CN=R3
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY)   subject  = /CN=hcc-briantest7
.unl.edu
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY)   err 20:unable to get local is
suer certificate
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) Tried to connect: -1
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) SSL: library failure: error:140
90086:SSL routines:ssl3_get_server_certificate:certificate verify failed
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) Round 3.
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) Send message (3).
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) Status (c: 3, s: 2)
07/11/23 22:26:19.814 (pid:1780000) (D_SECURITY) SSL Auth: SSL Authentication fa
iled
07/11/23 22:26:19.815 (pid:1780000) (D_SECURITY) AUTHENTICATE: method 256 (SSL)
failed.
07/11/23 22:26:19.815 (pid:1780000) (D_SECURITY) AUTHENTICATE: can still try the
se methods:
07/11/23 22:26:19.815 (pid:1780000) (D_SECURITY) HANDSHAKE: in handshake(my_meth
ods = '')
07/11/23 22:26:19.815 (pid:1780000) (D_SECURITY) HANDSHAKE: handshake() - i am t
he client
```

Compare output with debug disabled (left) versus enabled (right)

Identifier Mapping



Identity Mapping & Authorization

HTCSS has an ‘Identifier Mapfile’

- For identity-based schemes, the mapfile is an important tool to translate between authentication credentials and a HTCondor identifier.
- This mapfile has 3 columns: authentication method, authentication identifier, target HTCondor identifier.
  - By default, authentication identifier is a regexp.
- Sadly, this is known as the “CERTIFICATE_MAPFILE”.

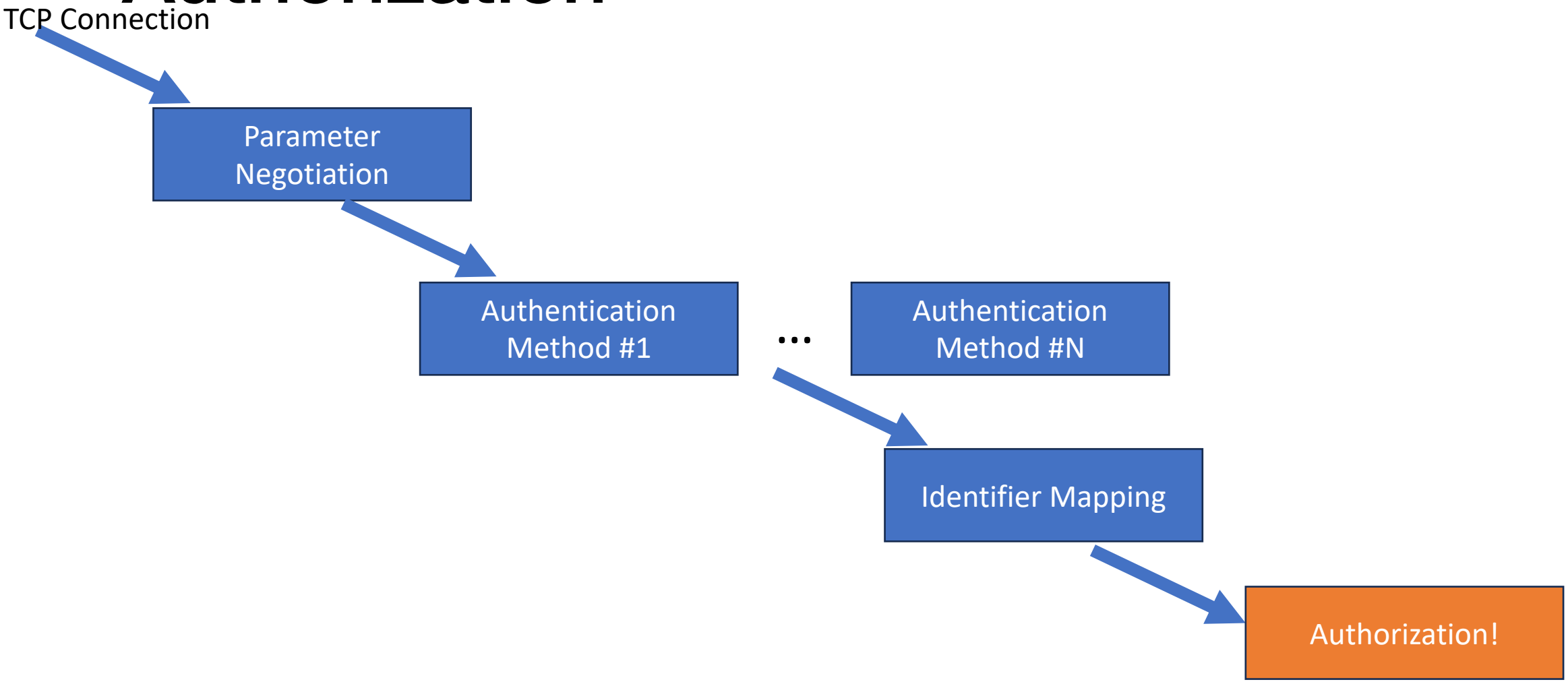
Identifier Mapfile Example

```
SCITOKENS "https://demo.scitokens.org" bbockelm@test.wisc.edu
SCITOKENS "https://wlcg.cloud.cnaf.infn.it/,27234843-fedf-42c8-bb81-
a1695bbd7c28" bbockelm@test.wisc.edu
SCITOKENS /^https\:\/\/osg\-htc\.org\/osdf,OSDF-(.*)@osg-htc.org$/ $1@osg-
htc.org
SSL "/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=bbockelm/CN=659869/CN=Brian
Paul Bockelman" bbockelm@test.wisc.unl.edu
INCLUDE /etc/condor/mapfile.d
```

Anonymous Identifiers

- If no HTCondor identifier can be established, protocols will default to an anonymous one.
- Unfortunately, we spell ‘anonymous’ as…
  - ‘unauthenticated@unmapped’ if no authentication is used.
  - ‘unauthenticated@unmapped’ if SSL is used but no client certificate is presented.
  - ‘ssl@unmapped’ if SSL is used, a client certificate is presented, but no mapping is available.
  - CONDOR_ANONYMOUS_USER@CONDOR_ANONYMOUS_USER if the ANONYMOUS method is used.
  - scitokens@unmapped if SCITOKENS is successful but no mapping is available.

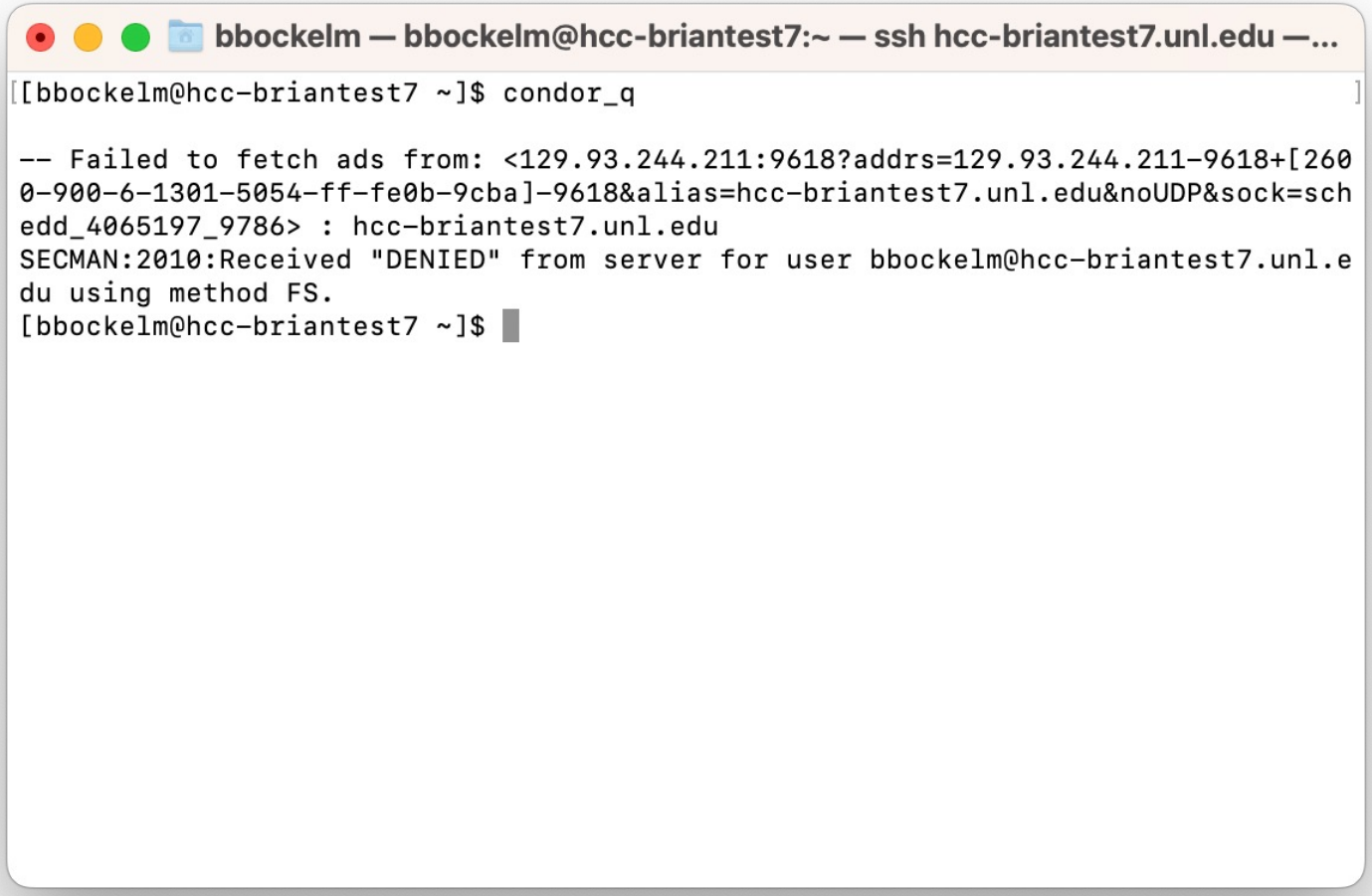
Authorization



Authorizations

- Once a HTCondor identifier is established, we finally determine whether the action is allowable.
- These are controlled by the ALLOW_* / DENY_* configurations.
- The ALLOW/DENY configurations are a list of identifiers of the form “$identifier/$host_restriction”. Examples:
  - *@wisc.edu/124.104.3.*
  - bbockelm@unl.edu
  - */*.wisc.edu
- DENY entries take precedence. If no matches for ALLOW, then the authorization is denied.

Authorization Failures



- Finally! We have an error message.

If in doubt – condor_ping it!

bbockelm — bbockelm@hcc-briantest7:~ — ssh hcc-briantest7.unl.edu — 103×17

```
[bbockelm@hcc-briantest7 ~]$ condor_ping -table ALL -type SCHEDD
          Instruction Authentication Encryption Integrity Decision Identity
                ALLOW             FS         AES       AES    ALLOW bbockelm@hcc-briantest7.unl.edu
                 READ           FAIL        FAIL      FAIL     FAIL FAIL  (use -verbose for more info)
                WRITE             FS         AES       AES    ALLOW bbockelm@hcc-briantest7.unl.edu
            NEGOTIATOR           FAIL        FAIL      FAIL     FAIL FAIL  (use -verbose for more info)
         ADMINISTRATOR           FAIL        FAIL      FAIL     FAIL FAIL  (use -verbose for more info)
                OWNER            FAIL        FAIL      FAIL     FAIL FAIL  (use -verbose for more info)
               CONFIG            FAIL        FAIL      FAIL     FAIL FAIL  (use -verbose for more info)
               DAEMON              FS         AES       AES    ALLOW bbockelm@hcc-briantest7.unl.edu
     ADVERTISE_STARTD              FS         AES       AES    ALLOW bbockelm@hcc-briantest7.unl.edu
     ADVERTISE_SCHEDD            FAIL        FAIL      FAIL     FAIL FAIL  (use -verbose for more info)
     ADVERTISE_MASTER              FS         AES       AES    ALLOW bbockelm@hcc-briantest7.unl.edu
[bbockelm@hcc-briantest7 ~]$
```

Final Thoughts / TODO list

- This was a fun overview to write!
- Please view this as a framework for understanding security handshake failures; impossible to enumerate all the possible reasons.
- Some observations from my side:
  - Incompatibility of settings is indecipherable from a network error.
  - The failure messages never say what part of the security handshake failed.
  - Authentication failure reasons are not in the failure messages, only the failure logs.
- Some great TODOs for the dev team!

Acknowledgements

This work is supported by NSF under Grant Nos. 2030508. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

iris hep

OSG

CHTC Center For High Throughput Computing