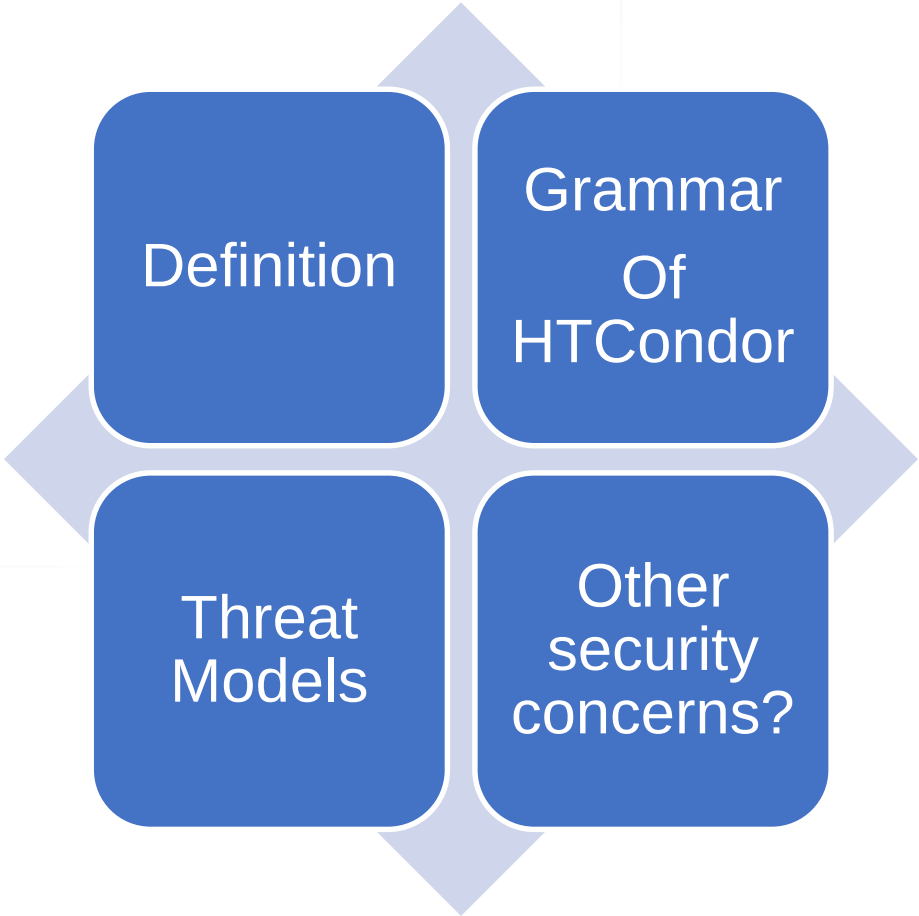
HTCondor Threat Models

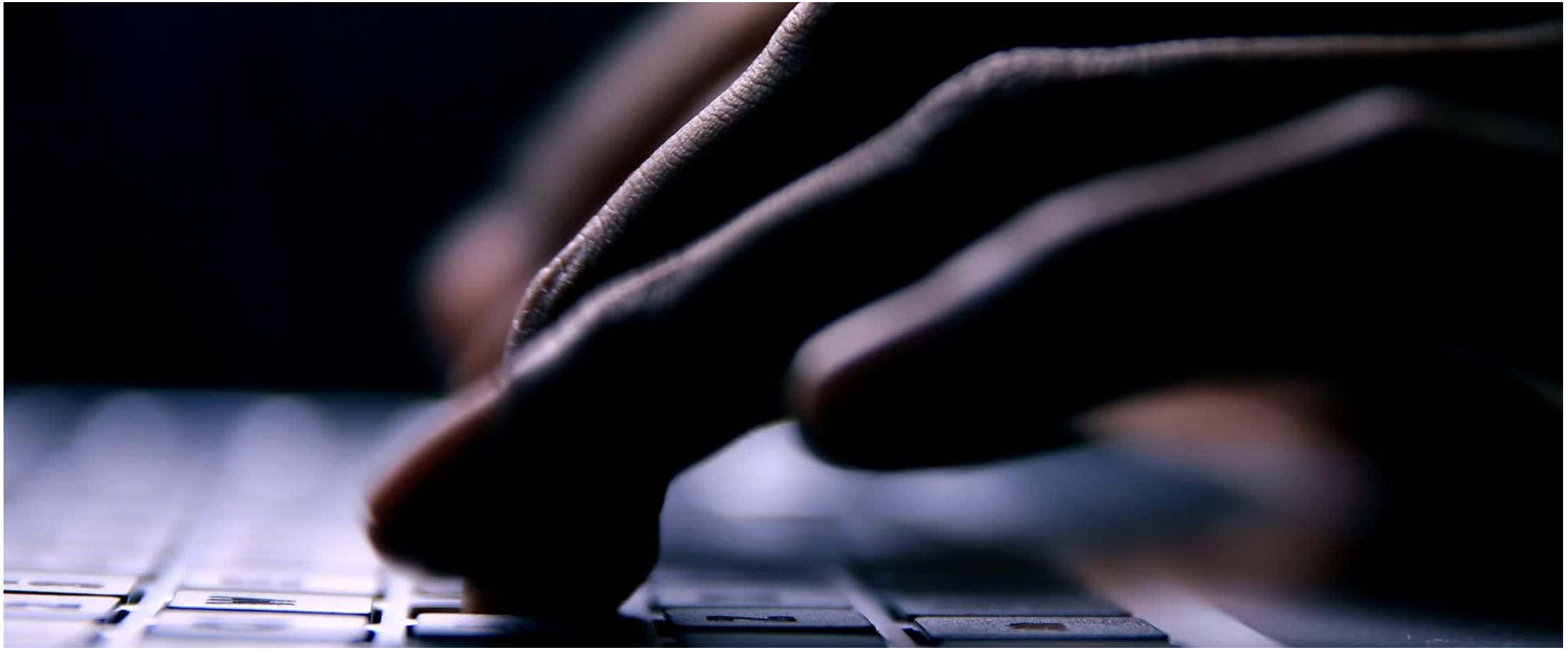
Greg Thain

Center for High Throughput Computing
Department of Computer Sciences
University of Wisconsin-Madison

Overview


Definition
Grammar Of HTCondor
Threat Models
Other security concerns?

Big Boss: "Make sure the pool is secure"



But what does "secure" mean?

- Site needs to decide what to prevent from happening
  - Depends on the site
    - Academic? Research? Commercial? Defence? Distributed? Local?

- The set of things you want to prevent is your "**Threat Model**"
    - And if maybe you want to log these things to prove that you are preventing

- Useful to think this through before trying to implement *anything*

"*If you don't know where you are going, any road will get you there.*"

Digression: Grammar of HTCondor

- Entities / Nouns / Classes
  - Jobs or Job Sets
  - Execute Points (worker nodes)
  - Users (submitters/owners)
  - Sandboxes / data
  - HTCondor services themselves

See the experimental "htcondor" tool for more

Threat model: Jobs

- Do we want to prevent some jobs from entering the system
  - First reaction: Is this based on "user identify / identification"?
    - Capability vs (Identity + Authorization map)
    - Bootstrap from some other security system?
      - Local identify
      - Munge
      - SSL, etc.
- Local vs. Remote?
- Does the content or type of the job matter?
- Access Point is the front door – where most of the checks are
  - Also be aware of the schedd's audit log
- Do we care about what a valid job does?
  - Better to prevent? Or better to check?

Threat model: Worker nodes / slots

- Do we want to prevent some worker nodes from "stealing" jobs?
  - Impossible in general to "prove" correct jobs execution
- Do we want to prevent a job from attacking the worker node?
- How do we do this in a glidein world?

Threat model: Other services

- Can we prevent a AP from joining the pool?
- Not just preventing an invalid job from being placed in a known AP?

Threat model: Users and groups

- If we are identifying users, are there some we want to
  - Prohibit from submitting jobs?
  - Prohibit from setting up new worker nodes
- Do we want to prevent users from running too many jobs?
- Do we want to tie a group to a job type
  - Analysis vs production? Multi-core vs single core?
- Do we want to prevent users from joining arbitrary groups?
- Do we want to limit where certain users' jobs run?

Threat model: Sandboxes and data

- Do we want to encrypt at rest the transferred sandbox?
- Do we need to encrypt the transfers on the wire?
- How do we authorize 3rd party transfers?
- Do we need integrity checks for transferred data

- Do we need to hide one job's sandbox from another

Threat model: Services themselves

- For a secure pool, 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
  - Munge
  - Pool Password (PASSWORD)
  - KERBEROS
  - SSL
  - IDTokens / SciTokens
  - ~~GSI~~

Thank you and questions

Thank you – Questions?

This work is supported by the NSF under Cooperative Agreement OAC-2030508. Any options, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.