Hall Pass: Per-action Authorization for Shared Agents

Alexander Zhang
azhang3@ucls.uchicago.edu
University of Chicago
Chicago, Illinois, USA

Alok Kamatar (advisor)
alokvk2@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Kyle Chard (advisor)
chard@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Abstract

Autonomous agents deployed across federated research infrastructure are most valuable when shared across teams, yet few frameworks provide a way to control who may invoke which of an agent's actions. An owner must choose between keeping a private copy of an expensive agent or exposing one that any caller can use, inspect, or terminate. We examine three models for authorizing calls to a shared agent and prototype the third in the Academy middleware, backing per-action permissions with Globus Groups and enforcing them on the message path. We demonstrate with a shared SAM3 agent on Purdue Anvil.

CCS Concepts

• **Computer systems organization** → **Distributed architectures**; • **Computing methodologies** → Distributed computing methodologies; • **Security and privacy** → *Access control.*

1 Introduction

Agentic systems are moving onto research infrastructure where agents drive instruments, simulations, and analysis across administrative boundaries. As science is interdisciplinary and multi-institutional, sharing agents between researchers is necessary to enable collaboration and progress. In existing frameworks, an agent inherits a user's full permissions, making a compromised agent far worse than a single bad request, and many current frameworks authenticate the user without identifying the agent at all. We explore three models for authorizing agent communication based on existing OS access-control models [4] and examine the tradeoffs between them. We instantiate the per-action model in Academy [3], an open-source middleware for building agents on research infrastructure, analyze its overhead, and illustrate how it behaves in a practical scenario.

2 Three Models for Authorizing a Shared Agent

In Academy [3], agents communicate through a mailbox model: an agent sits behind a mailbox addressed through a handle, exposing actions—methods invocable remotely—and every message passes through an exchange that routes, queues, and delivers it—a natural place to enforce access (Figure 1). We experiment with three models of authorization that provide different granularity and overhead (Figure 2). We implement the third as an extension to Academy's exchange and agent runtime.

Identity comes from Globus Auth [5]: the exchange derives a caller's group memberships server-side from its token, so a sender cannot self-report them [2].

The agent is the user. In the simplest model the agent carries its launcher's credentials as a delegated OAuth token, a standard protocol for authenticating cloud services; as under Linux discretionary access control, the agent simply acts as the user. Nothing new must be deployed, but a compromised agent is a compromised user, and sharing the agent means handing out the user's full authority.

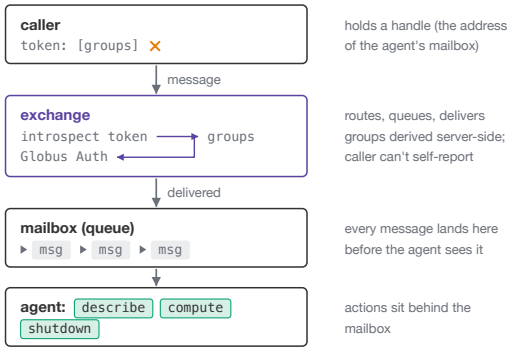


Figure 1: Architecture of communication through the Academy exchange.

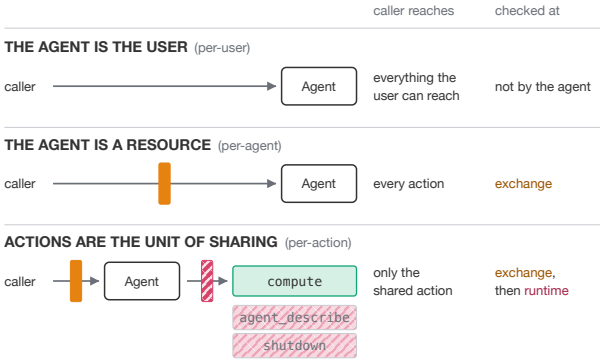


Figure 2: Different granularities for permissions and sharing between agents.

The agent is a resource (per-agent). The next model gives the agent its own identity and puts admission control at its mailbox. Admission, though, is all or nothing: a caller who may reach the mailbox may invoke every action behind it, including shutdown. Academy already implements this model, and its Globus exchange registers each agent as a Globus Auth resource server [5]. Permission to write to the mailbox is enabled by allowing a Globus group access to that mailbox. All permissions in this model are enforced at the exchange level; any requests that are allowed to be written to the mailbox are processed.

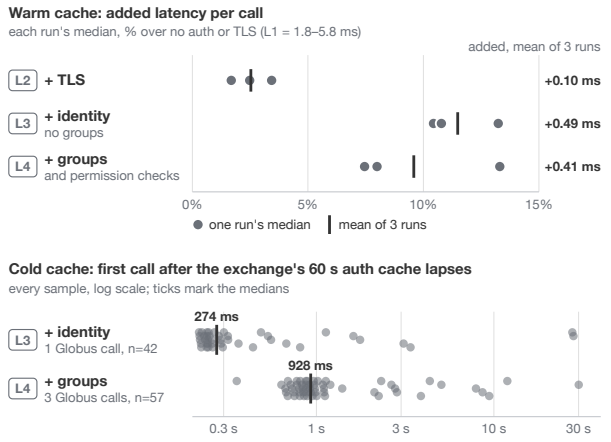


Figure 3: Added per-message latency across the four exchange configurations, L1–L4, on one machine over loopback.

Actions as the unit of sharing (per-action). Finally, the last model keeps admission at the mailbox and pushes authorization down to individual actions. The owner declares, per action, which groups may invoke it, and everything unshared stays closed. Each message is checked at two points: the exchange intersects the caller's groups with all groups that may invoke any action on the receiver, without introspecting the message, and after delivery the receiver checks them against the groups for the specific action called.

3 Evaluation

We first evaluate the overhead of authorization, then demonstrate potential resource advantages of a shared agent for a realistic application.

3.1 Overhead

To evaluate the overhead of each of the authorization methods, we measure the end-to-end latency of passing a message across four configurations of the same exchange, L1–L4: unauthenticated, +TLS, +identity (no group lookup), and +groups and permission checks (Figure 3). We compare the overhead incurred by the authorization to the overhead of ordinary TLS. Both tokens and group memberships are cached, so overhead splits into a warm and a cold case.

With a warm cache, TLS adds 0.10 ms and identity a further 0.40 ms; groups and per-action checks add nothing measurable beyond that—the 0.09 ms between L3 and L4 is inside block-to-block noise—so the full stack costs 0.41 ms on a 4.14 ms mean baseline (1.8–5.8 ms across blocks). In the case where the identity is not in the cache, identity alone and identity with groups take a median 274 ms and 928 ms respectively, the gap reflecting the additional group lookups.

3.2 SAM3 Demo

Scientists using experimental facilities like the Advanced Light Source (ALS) transfer and analyze their data on supercomputers. Many experiments (by different users) could benefit from similar analysis procedures. A shared "analysis" agent could accumulate collective knowledge of techniques used to derive insights from ALS images. In addition, a shared agent alleviates the burden of allocating resources and loading analysis tools for each user.

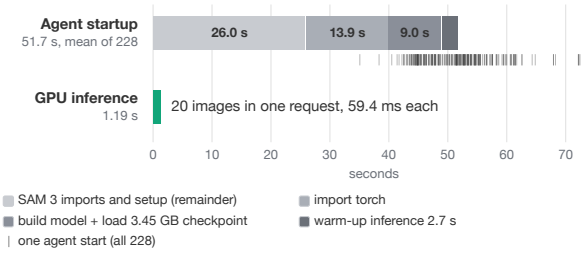


Figure 4: SAM3 agent startup versus useful work.

As a proof of concept, we deployed an agent with the SAM3 model [1] to segment the same fixed box in 20 pictures on an A100 at Purdue Anvil. Each took 59.4 ms; reaching the first one took 52 seconds (Figure 4). Twenty users, each with their own GPU job and each running the same 20-image workload, spend 19 minutes of GPU time to perform 24 seconds of segmentation. Sharing one agent does the same work in 89 seconds of GPU time. Both setups ran with authorization on, so these times already include it. Every client signed in as one Globus identity.

4 Conclusion

Safe sharing comes down to what the authorization system treats as the principal. Making the action the unit of sharing resolves the dilemma at a warm cost of about half a millisecond per call. Measuring at production scale remains future work.

Acknowledgments

Anvil access was provided through the NSF ACCESS program under allocation CIS250223, "Multi-site serverless scheduling for science." Claude (Anthropic) was used in preparing select figures and in checking the text.

References

[1] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. 2025. SAM 3: Segment Anything with Concepts. *arXiv preprint arXiv:2511.16719* (2025). doi:10.48550/arXiv.2511.16719

[2] Globus. [n. d.]. Globus Groups API. https://docs.globus.org/api/groups/. Accessed August 6, 2026.

[3] Alok Kamatar, J. Gregory Pauloski, Yadu Babuji, Ryan Chard, Mansi Sakarvadia, Daniel Babnigg, Kyle Chard, and Ian Foster. 2026. Empowering Scientific Workflows with Federated Agents. In *Proceedings of the 2026 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 1403–1418. doi:10.1109/IPDPS65963.2026.00114

[4] Butler W. Lampson. 1974. Protection. *ACM SIGOPS Operating Systems Review* 8, 1 (1974), 18–24. doi:10.1145/775265.775268

[5] Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. 2016. Globus Auth: A Research Identity and Access Management Platform. In *Proceedings of the 2016 IEEE 12th International Conference on e-Science*. IEEE, 203–212. doi:10.1109/eScience.2016.7870901