

SECURITY ENGINEERING

The Secret-Free Fleet

A Security-First Approach to JIT Admin
and MDM Automation

Justin Gianelloni • Matt Parker

Why we're here



We built a way to give Macs **admin access, managed locally and by our IDP, without putting any secrets on them.**



The system built outlines a **reusable architectural pattern,** not just a JIT admin tool.

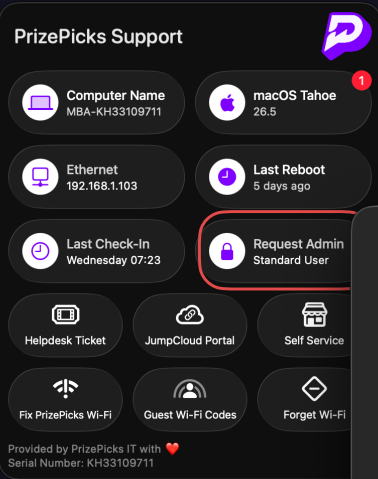
TRANSPARENT TO THE USER

User Focused First

1

User opens requests admin elevation

Selects a reason for access





Would you like to request administrator privileges?

You are currently a standard user. To request administrator privileges, please select a reason below.

Administrator privileges expire in 20 minutes.

All actions during this time will be logged.

Reason *

* Required Fields

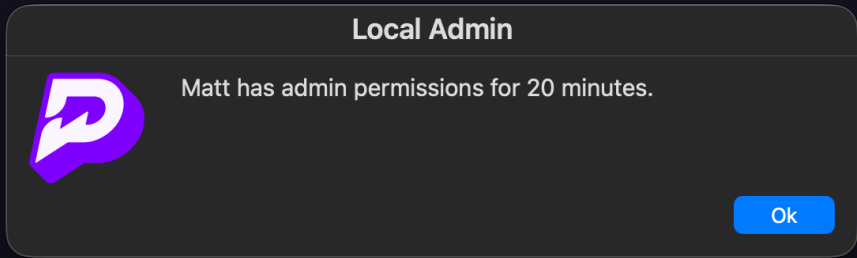
Cancel

Request Privileges

2

Admin rights granted

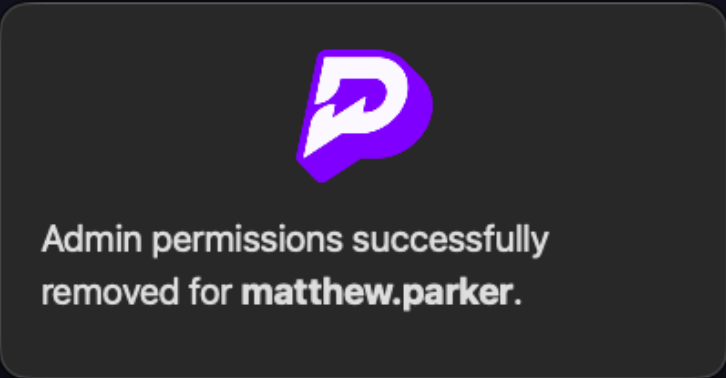
20-minute window, immediately active



3

Auto-revoked + logged

Every action captured, shipped to SIEM



No tickets. No standing access. Every action logged.

THE THESIS

How do you give a fleet **privileged
directory access** without
distributing the keys?

PRIZEPICKS

1

ACT ONE

The Problem

THE PROBLEM

The shape of the problem

Every fleet automation that touches your directory has the same shape:

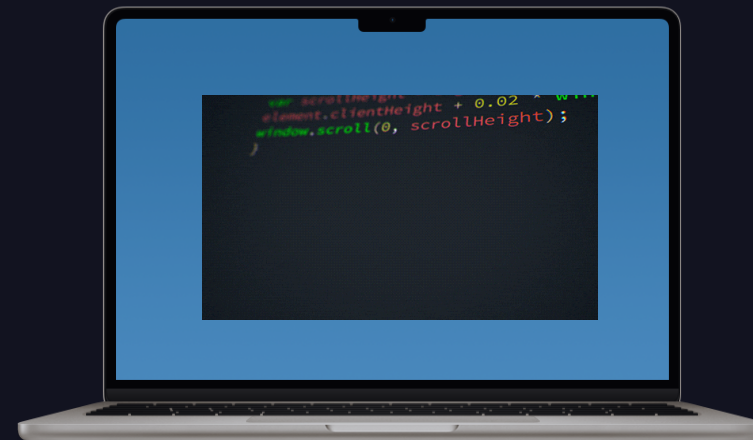
Endpoint holds credential

The secret lives on disk, in memory, or in config



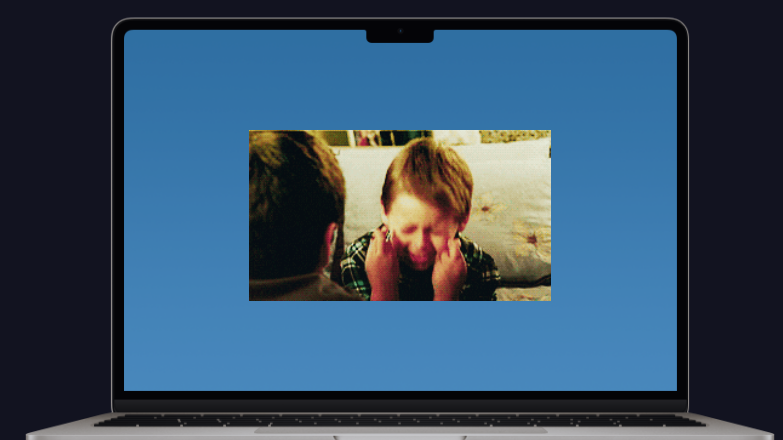
Endpoint calls API

Uses the credential to authenticate



You hope nothing goes wrong

Because revocation and rotation rarely happen



This isn't an MDM problem or a IdP problem. **It's the default architecture.**

THE PROBLEM

What standing secrets on endpoints actually cost you

- Rotation is **fleet-wide and rarely done**
- Compromise of one endpoint = **compromise of the credential**
- Audit trail lives on the endpoint, which is **the thing you don't trust**
- Manual revocation: it's "scheduled," it's "tracked," **it doesn't happen**



THE PROBLEM

The two half-solutions everyone tries

"JUST DON'T GIVE ADMIN"

Works until it doesn't. Users find workarounds, tickets pile up.

Credential still exists somewhere. Users route around it.

"TEMPORARY ADMIN SCRIPTS"

Better, but the script still holds the API token.

Revocation is best-effort. Token rotation is manual.

Both leave the credential on the endpoint. Both leave revocation as a hope.

PRINCIPLES

REFRAMING

WRONG QUESTION

"How do we rotate the secrets faster?"

RIGHT QUESTION

"What if the endpoint never held the secret?"

Let's review the system that makes this possible.

2

ACT TWO

The Architecture

Meet Hookshot

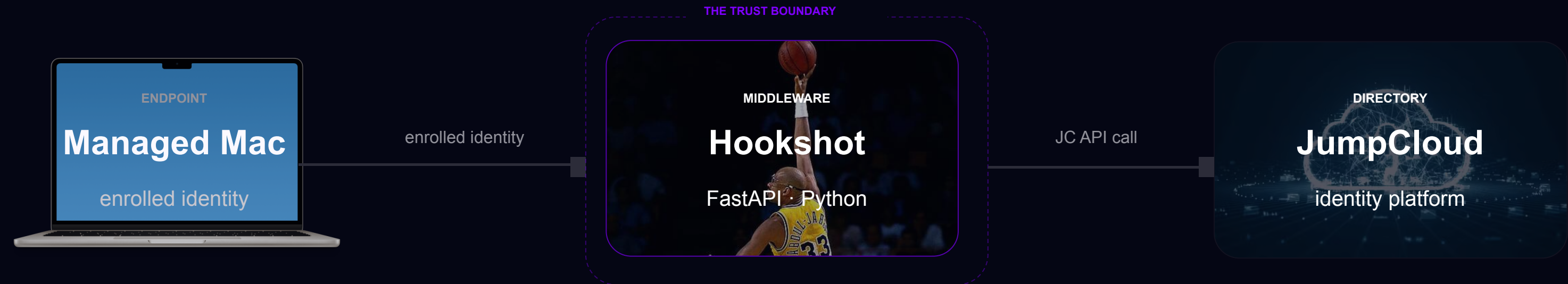
Custom middleware API

- Sits between managed Macs and JumpCloud
- **Endpoints never hold Directory tokens** 🔑
- Single authorized interface for any privileged directory operation



THE ARCHITECTURE

Architecture overview



Hookshot is the **only** entity holding the **Directory credential**. No endpoint ever touches it.

PRIZEPICKS

THE ARCHITECTURE

How the Mac proves it's allowed to ask

TODAY

- Server-side validation of enrolled device identity
- Trust established at MDM enrollment
- No rotatable shared secret stored on disk

IN PROGRESS

- SCEP-issued device certificates via MDM
- Hookshot validates cert against enrollment CA
- Cryptographic proof of enrollment, not a password

The property: the endpoint proves it's an enrolled device **without holding a rotatable shared secret.**

Why FastAPI

ASYNCH REQUEST HANDLING

The JIT flow feels synchronous to the user, even though parts of it are queued behind the scenes.

```
@router.get("/get_user_id/{username}", status_code=200)
async def get_user_id(username: str, services: ServicesDepends) -> UserID:
    """
    Look up a JumpCloud user ID by username.

    Returns the JumpCloud `user_id` for the given `username` path parameter.
    """
    try:
        user_id: str = await services.jumpcloud.get_user_id(username)
    except Exception as e:
        raise HTTPException(500, "An unknown error has occurred.") from e
    return UserID(user_id=user_id)
```

AUTO-GENERATED OPENAPI DOCS

Every allowable fleet action is enumerated automatically. Audit by default, not by effort.

POST

/v1/macos/update_access_request

DELETE

/v1/macos/revoke_sudo

TYPE-CHECKED MODELS

Reduces a class of bugs we especially don't want in a security-critical code path.

```
class AccessRequest(BaseModel):
    user_id: str
    system_id:
    str remarks: str
```

The OpenAPI surface as audit

METHOD	ENDPOINT	DESCRIPTION
POST	/v1/macos/update_access_request	Request temporary local admin elevation
DELETE	/v1/macos/revoke_sudo	Revoke active elevation for device
GET	/v1/macos/sudo_status	Query current elevation status

If it's not in OpenAPI, **the fleet can't do it.**

Scope enforcement

A machine can request elevation for *itself*, not arbitrary directory operations.

- Hookshot validates: **who is asking, what they're asking for**, whether their identity is allowed to ask
- Attack surface exposed to any individual endpoint is intentionally narrow
- A compromised Mac can request its own elevation: **same blast radius as a stolen admin password**



Dual-control security model

LOCAL ENFORCEMENT

- sudoers entry written
- Admin group membership added
- LaunchDaemon timer armed

HOOKEHOT TO JUMPCLOUD ACCESS REQUEST API

- Access request logged server-side
- JumpCloud validates and acknowledges
- Server-side revocation available

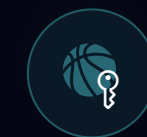


LOCAL KEY



Managed Elevation

LOCKED · SERVER-VERIFIED



HOOKEHOT KEY

- Either key can **grant**
- Either key can **revoke**

Two independent systems know the elevation is valid. Either side can revoke.

PATTERN, NOT TOOL

Hookshot is a pattern, not a tool.

Anything that today requires a Directory token on the endpoint can move behind this layer.

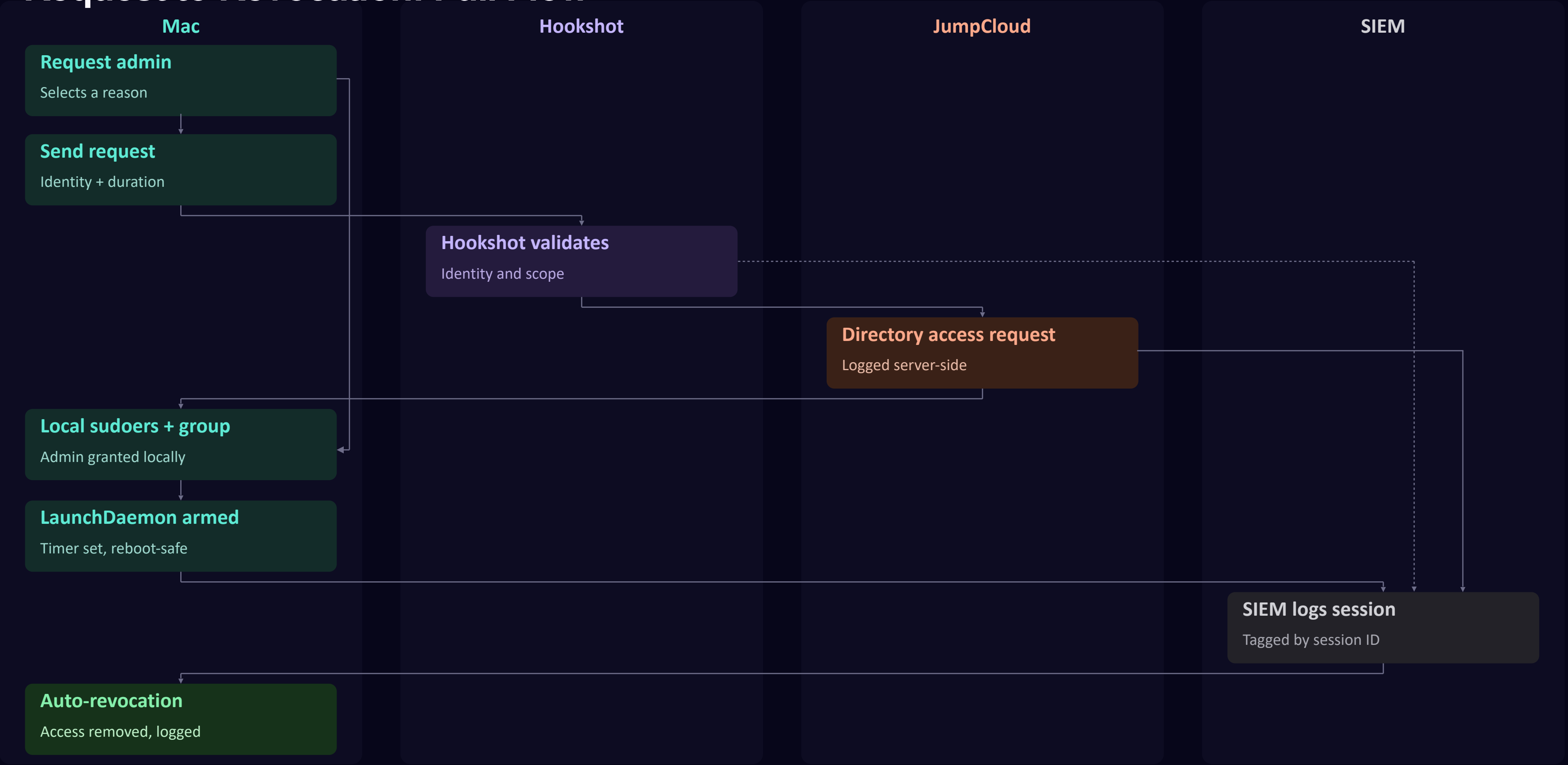
Back to the endpoint

3

ACT THREE

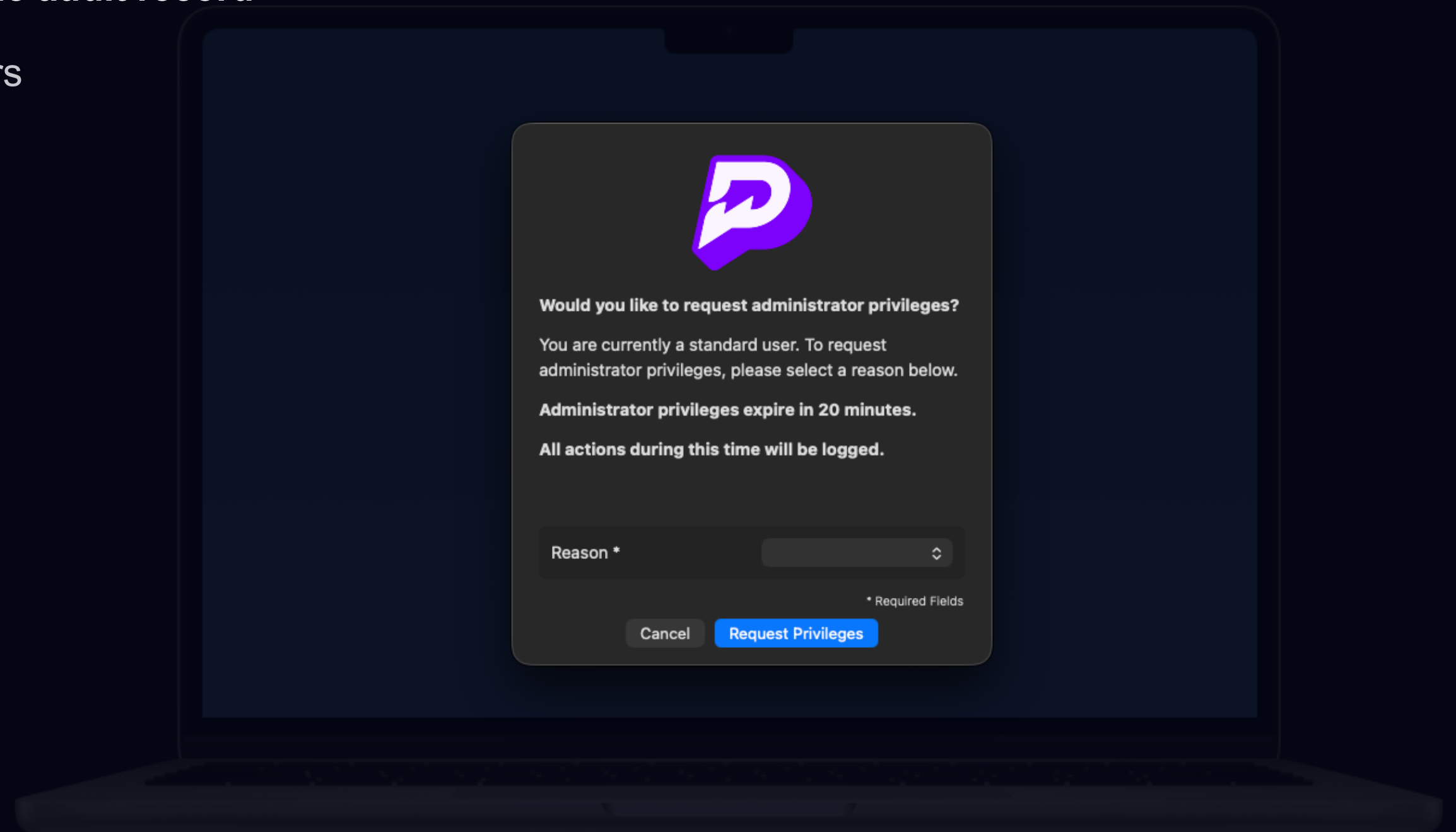
The JIT Flow

Request to Revocation: Full Flow



Steps 1–2: SwiftDialog UI

- Native macOS feel
- Reason field is required: **becomes part of the audit record**
- Fully scriptable with SwiftDialog parameters



Step 3: The request to Hookshot

```
# Sent from managed Mac, no secret in the request

curl -X POST https://hookshot.internal/v1/macos/grant_sudo \
  --cert "${CERT}" \
  -H "Content-Type: application/json" \
  -d '{"reason": "Installing dev tooling for project X", "duration": 1200}'
```



No credential in the request. Identity is proven by the client certificate, not a shared secret.

Step 4: What Hookshot does

- 1 Validates the requesting identity against MDM enrollment
- 2 Confirms the machine is allowed to request its own elevation
- 3 Calls JumpCloud access request API on behalf of the machine
- 4 Returns approval response: Mac proceeds with local mutation

Step 5: Local sudoers and group mutation

```
# On request  
dscl . -append /Groups/admin GroupMembership "$USERNAME"  
echo "$USERNAME ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/localadmin-jit
```

Why both group membership AND sudoers? macOS treats them as separate authorization paths. Belt and suspenders.

Step 6: LaunchDaemon-driven revocation

Local Enforcement

Armed with a root-level LaunchDaemon delay trigger that survives reboots, clears local admin groups, and deletes the custom sudoers file.

```
<key>StartInterval</key>
<integer>1200</integer>
<key>ProgramArguments</key>
<string>/usr/local/bin/revoke-localadmin</string>
```

Server-Side Fallback

JumpCloud independently triggers an automated server-side revoke command at the end of the window.

- **The Payoff:** Belt-and-suspenders automation. If local enforcement fails or goes offline, the directory API cuts access from above.

Step 7: Session logging via SIEM

SIEM captures all process, file, and auth events during the elevated window.

```
# SIEM ProcessRollup2 event (during elevated session)

UserName: matthew.parker

CommandLine: /usr/local/bin/brew install node

Tags: jit_session_id=abc123  reason="dev tooling project x"
```

Captured by user, by session, with the original reason text. **Scoped to the elevated window.**

Step 8: Session data in your SIEM

```
{  
  "event_type": "jit_admin_session" ,  
  "host":      "macbook-jg-01" ,  
  "user":      "justin.gianelloni" ,  
  "session_id": "abc123" ,  
  "reason":    "Installing approved dev tooling"  
}
```

SOC PIVOT

"Show me all elevations this week"

↓

"Show me every command run by user X during their elevation on machine Y"

THE JIT FLOW

The MDM side: configuration profiles and PPC

PPPC PROFILE

Full Disk Access for SIEM sensor (TCC)

CLIENT CERT DELIVERY

Config profile for Hookshot client cert / enrollment trust

LAUNCHDAEMON MANAGEMENT

Privileges configured for daemon's launchd context

SIEM SENSOR CONFIG

Ensure process and auth telemetry enabled for elevated sessions

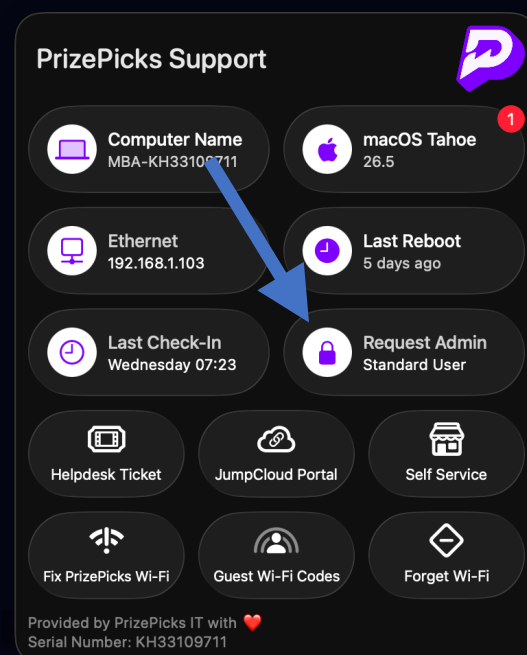
Without these, the whole thing fails silently in the worst possible way.

User clicked a button.

No secret left the endpoint.

Every action is logged.

Revocation happened automatically.



```
# Sent from managed Mac, no secret in the request
curl -X POST https://hookshot.internal/v1/macOS/grant_sudo \
  --cert "${CERT}" \
  -H "Content-Type: application/json" \
  -d '{"reason": "Installing dev tooling for project X", "duration": 1200}'
```

```
{
  "event_type": "jit_admin_session" ,
  "host":      "macbook-jg-01" ,
  "user":      "justin.gianelloni" ,
  "session_id": "abc123" ,
  "reason":    "Installing approved dev tooling"
}
```



Admin permissions successfully
removed for **matthew.parker**.

4

ACT FOUR

Evolution & In-Flight Changes

From Hackathon to Production

The MVP Origins

- **Fast-Paced Prototype:** Built as a rapid hackathon project to solve a burning, immediate pain point for our deployment.
- **Monolithic Assumptions:** Designed around a straightforward script-and-server proof-of-concept to see if Hookshot was viable.

The Reality Check

- **Enterprise Posture:** Re-architected into critical corporate infrastructure requiring strict operational uptime, monitoring, and observability.
- **The Modular Win:** Because the initial design was strictly modular, we were able to completely migrate core back-end components without disrupting the client-side experience.

Current In-Flight Upgrades

Cryptographic Identity

- **mTLS Authentication:** Transitioning from basic server-side validation of enrolled device identity to true mutual TLS.
- **Device Certificates:** Leveraging MDM to issue SCEP device certificates.
- **The Win:** Hookshot validates requests directly against your enrollment CA, guaranteeing cryptographic proof of enrollment without a rotatable shared secret on disk.

Infrastructure & GitOps

- **Automated Pipelines:** Moving away from manual deployment pieces to a fully automated, state-driven GitOps workflow.
- **Deep Monitoring:** Implementing real-time container health checks, uptime tracking, and alerting metrics.
- **The Win:** Treating the Hookshot middleware as Tier-1 enterprise infrastructure.

Telemetry Evolution

- **SIEM Provider Migration:** Actively swapping out the underlying SIEM provider back-end architecture.
- **Decoupled Pipelines:** Because Hookshot was built with a decoupled model, logging endpoints remain identical even as data destinations shift.
- **The Win:** Zero downtime or disruption to security analytics during the migration.

The Hard Truths

Velocity vs. Security

- **The Reality:** FastAPI gave our team incredible developer velocity and self-documenting code models out of the gate.
- **The Shift:** The real engineering lift wasn't writing the API endpoints—it was hardening the container, narrowing the API surface area, and managing the security posture around it.

POST

/v1/macos/update_access_request

DELETE

/v1/macos/revoke_sudo

Lean on Native Frameworks

- **The Reality:** Never try to roll your own security or trust scripts when native platforms exist.
- **The Shift:** Trust and exploit built-in, load-bearing native frameworks (like Apple's Endpoint Security subsystem via eslogger and MDM-driven SCEP profiles) rather than trying to script custom validation mechanisms.

```
es_client_t *client;
es_new_client_result_t result = es_new_client(
    &client, ^(es_client_t *c, const es_message_t *msg) {
        printf("Event: %d\n", msg->event_type);
    }
);
```

5

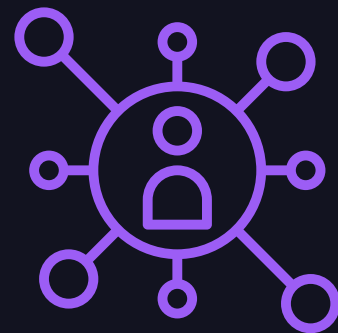
ACT FIVE

Beyond JIT

What else fits the pattern

GROUP MEMBERSHIP ON DEMAND

Dynamically add a device or user to a sensitive project group in JumpCloud without leaving a wide-open write token on the local machine.



SELF-SERVICE BIND RESETS

Offload safe, self-service directory tasks to the middleware layer.



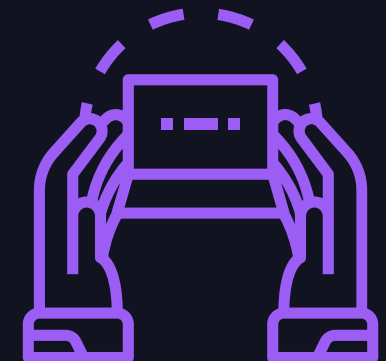
MFA PROXYING

Proxy complex JumpCloud MFA enrollment and verification calls securely through Hookshot, treating the local device as a request channel, not a trusted endpoint.



ASSET MANAGEMENT

Update device records, custom attributes, or assignment data in JumpCloud through Hookshot. The endpoint reports what it knows; Hookshot decides what gets written.



Any privileged directory op that today needs a token on the endpoint. Trusted endpoints can't be treated the same as a trusted server.

What you don't get for free

Tier-1 Availability

- **The Operational Reality:** Hookshot is now critical, load-bearing enterprise infrastructure.
- **The Tradeoff:** It requires the exact same high-uptime, monitoring, and alerting posture as your MDM. If the middleware is down, your dynamic fleet capabilities stall.

OpenAPI Governance

- **The Operational Reality:** Your auto-generated OpenAPI surface dictates exactly what actions the fleet can request.
- **The Tradeoff:** This layer is your true audit and attack surface. Every new endpoint or method you add explicitly expands what a compromised fleet device could theoretically execute. You must keep it ruthlessly tight.

Load-Bearing PKI

- **The Operational Reality:** The entire "secret-free" security model relies on client certificate validation.
- **The Tradeoff:** Certificate lifecycle management is no longer a set-and-forget background process. An expired or broken CA pipeline instantly triggers a production-level fleet outage.

Strict Dev/GitOps Overhead

- **The Operational Reality:** Moving away from a collection of simple scripts means adopting a formal software development lifecycle.
- **The Tradeoff:** You trade simple "script adjustments" for mandatory GitOps deployment testing, container vulnerability scanning, and framework maintenance to avoid introducing bugs into a critical security path.

CLOSE

Three takeaways

Structural Flaw

Storing secrets on endpoints are an architectural problem, not a hygiene problem.

Inversion of Trust

A custom middleware layer turns the endpoint into a requester instead of a holder.

The Extensible Blueprint

JIT admin is one application. The pattern generalizes.

Resources

REPO

THE CODE STATUS



Repository: Inner-source / code not yet approved for public release.

KEEP THE CONVERSATION GOING

MACADMIN SLACK



Handles: @Justin Gianelloni, @mattparker



Find us in the hallway afterward, we'd love to look at your architecture or share sanitized configurations.

Questions?

Justin Gianelloni • Matt Parker

MacAdmins Penn State 2026

PRIZEPICKS