

SeatPrune: Enterprise Security & Governance Whitepaper

Executive Security Overview

SeatPrune is an enterprise-grade autonomous SaaS license reclamation and spend governance agent built for Security, IT, and FinOps teams. It identifies inactive, orphaned, and zombie user licenses across high-cost developer tools (including GitHub Enterprise, GitHub Copilot, and Slack Enterprise Grid), computes exact wasted spend, dispatches automated user check-in alerts, and executes safe seat revocation.

Because SeatPrune interacts directly with enterprise identity systems and developer tool licensing APIs, security and execution safety are built into its core architecture.

1. Execution Safety & Dry-Run Locks

SeatPrune implements mandatory safety controls to prevent accidental user de-provisioning:

- **Default Dry-Run Execution:** All reconciliation commands (`seatprune prune` , `execute_reclamation`) default to `--dry-run = True` . Destructive modifications require explicit, intentional flag overrides (`--no-dry-run`).
 - **Simulation Mode & Audit Emulation:** In dry-run mode, API calls simulate revocation actions, calculating exact cost recoveries and logging proposed changes without mutating upstream permissions.
 - **Granular Reclaimer Isolation:** Revocation tasks are evaluated individually per connector instance. If a single user de-provisioning fails, the transaction is isolated without corrupting state or impacting other organization members.
-

2. Protected Account Governance & Allowlisting

To protect critical executive, bot, automation, and core infrastructure accounts:

- **Pydantic Allowlist Enforcer:** SeatPrune maintains a strict allowlist configured via `SEATPRUNE_ALLOWLIST` (or `config.py`).
 - **Automated Exemption Checks:** Prior to evaluating inactivity thresholds or executing seat reclamation, candidate user accounts are validated against the allowlist.
 - **Bypass Logs & Transparency:** Excluded accounts (e.g., CI/CD bot accounts, C-suite executives, system integrators) generate explicit `allowlist_protected` audit events in reclamation logs.
-

3. Credential Isolation & Token Security

SeatPrune follows zero-trust credential handling principles:

- **Environment Variable Injection:** Authentication credentials (`GITHUB_TOKEN` , `SLACK_WEBHOOK_URL`) are read strictly from process environment variables or securely stored `.env` files via Pydantic Settings. Credentials are never written to disk or hardcoded.
- **Least-Privilege Token Scopes:**
 - **GitHub REST/GraphQL Token:** Requires minimum read-only scope for auditing (`admin:org` read for member activity, `manage_billing:copilot` for Copilot seats). Revocation requires scoped write access (`admin:org` / `manage_billing:copilot`).

- **Slack Webhooks:** Uses incoming webhooks with restricted channel posting privileges without requiring global workspace admin tokens.
 - **Zero Log Leakage:** Authorization headers, token signatures, and private secrets are masked across all stdout logs, FastMCP resources, and stderr outputs.
-

4. Audit Logging & Compliance Standards

SeatPrune maintains full auditability for compliance frameworks (SOC 2 Type II, ISO 27001, HIPAA):

- **Structured JSON Audit Logs:** Audit summaries and execution histories are serialized into structured JSON conforming to strict Pydantic schemas (`InactivityReport` , `ReclamationPlan` , `ExecutionResult`).
 - **FastMCP Inspection Resource:** Real-time audit states are exposed via read-only FastMCP resources (`seatprune://audit/latest`), allowing security operations centers (SOC) and SIEM tools to ingest telemetry safely.
 - **Deterministic Financial Calculations:** Inactivity evaluations adhere to fixed financial cost matrices (\$19/mo per Copilot seat, \$21/mo per GitHub Enterprise seat) to ensure deterministic calculation of monthly and annual savings.
-

5. Local State Containment & Data Privacy

- **Zero External Data Storage:** SeatPrune operates locally within your infrastructure or CI runner. No organizational metadata, employee activity logs, or credentials are transmitted to third-party SaaS servers.
 - **Ephemeral In-Memory Processing:** User activity data gathered during REST/GraphQL API queries is processed ephemerally in memory and discarded upon completion of the execution cycle.
-

6. Security Checklist for Production Deployment

Before deploying SeatPrune to production:

- ☒ Verify `SEATPRUNE_ALLOWLIST` contains all service accounts, bots, and executive emails.
- ☒ Confirm `GITHUB_TOKEN` is issued with minimal necessary OAuth/PAT permissions.
- ☒ Test new reclamation rules using `seatprune prune --dry-run` or `execute_reclamation(dry_run=True)` .
- ☒ Integrate FastMCP server over secure stdio transport with your internal IDE/agent infrastructure.