Chris Chevalier product@chrischevalier.com

Application Control Case Study

How Trust Engine increased application control adoption by 400%

Executive summary

I redesigned an application control product that had stalled adoption because it authorized executables only by hash values. That approach required preauthorizing every patch and update, creating heavy administrative overhead and frequent user disruption. By introducing the **Trust Engine**, a set of complementary trust mechanisms that give every executable "a reason to run", the team reduced operational friction, preserved a strict default-deny posture, and drove a **400% increase in adoption** across target environments.

Background and problem statement

The original product enforced execution by matching file hashes. Any change to an executable such as patches, rebuilds, or updates produced a new hash and therefore required reauthorization. This produced three predictable problems:

- High administrative cost: Security teams and helpdesk staff spent disproportionate time re-hashing and whitelisting files.
- Slowed updates: Rollouts were delayed because updates had to be pre-hashed and whitelisted, increasing exposure windows.
- User disruption: Legitimate updates and day-to-day changes were blocked, generating tickets and frustration.

Because of these constraints, application control was typically limited to low-change environments (for example, SCADA systems) and rarely deployed on workstations where software changes are frequent.

Solution overview

The Trust Engine reframes the problem: instead of asking “Is this exact file hash authorized?” it asks “Does this executable have a verifiable reason to run?” The Trust Engine combines multiple, auditable trust signals so administrators can express intent at the right level of granularity—publisher, updater, path, user, or the machine’s current state—rather than at the brittle file-hash level.

Design goals:

- Preserve a **default-deny** security posture.
- Minimize day-to-day administrative work.
- Enable safe, auditable exceptions for legitimate change.
- Make app control practical for high-change endpoints like workstations.

The Trust Engine: mechanisms and rationale

Easy Lockdown

The agent scans the endpoint and whitelists everything already installed.

- **Why it matters:** Quickly establishes a known, stable baseline without requiring pre-authorization of every patch level.
- **When to use:** Initial deployment and for endpoints where installed software is trusted or existing risk is acceptable.
- **Operational effect:** Dramatically speeds rollout without increasing ticket volume.

Trusted Publisher

Administrators mark a certificate publisher as trusted. Any executable signed by that publisher is allowed.

- **Why it matters:** Eliminates per-file authorization for widely trusted vendors (for example, Microsoft) and internal signing CAs.
- **When to use:** Enterprise vendors, ISVs, and internal code signing authorities.
- **Operational effect:** Reduces whitelist churn while maintaining cryptographic assurance.

Trusted Updater

When an application or service is designated as a Trusted Updater, any executable it installs is allowed.

- **Why it matters:** Supports auto-updaters so user-initiated updates don't break execution.
- **When to use:** Applications with controlled updater processes (browsers, collaboration tools, managed agents).
- **Operational effect:** Enables seamless updates without manual pre-authorization.

Trusted Path

Administrators designate a filesystem path where any executable is permitted to run.

- **Why it matters:** Supports developers and helpdesk staff who frequently replace or run utilities.
- **When to use:** Developer machines, support workstations, or controlled shared folders.
- **Operational effect:** Reduces friction for legitimate, frequent changes while keeping other locations locked down.

Trusted Person

Administrators mark specific user accounts as trusted. When a Trusted Person runs an unauthorized executable they are prompted to approve it, and that approval is auditable.

- **Why it matters:** Empowers knowledgeable users to resolve issues quickly while preserving an audit trail.
- **When to use:** Admins, developers, and power users.
- **Operational effect:** Lowers helpdesk load and speeds problem resolution without broad policy relaxations.

Implementation approach

1. **Pilot selection:** Choose a representative mix of endpoints—developers, helpdesk, and standard workstations—to validate policies and telemetry.
2. **Agent rollout with Easy Lockdown:** Deploy the agent and trigger Easy Lockdown to scan and create a local whitelist to establish a trusted baseline quickly.

3. **Configure trust policies:** Add Trusted Publisher entries for major vendors and register Trusted Updaters for common auto-updaters.
4. **Define Trusted Paths and Persons:** Create safe paths for developers and mark a small set of trusted users.
5. **Monitoring and audit:** Log every trust decision, review new authorizations regularly, and integrate with change control.
6. **Iterate and scale:** Use pilot telemetry to refine rules, then expand across the fleet.

This phased approach minimized disruption, produced measurable telemetry early, and allowed policy tuning before broad rollout.

Results and metrics

- **Adoption:** Adoption increased by **400%** in targeted segments after rollout.
- **Operational efficiency:** Helpdesk tickets and pre-authorization tasks dropped significantly. Update rollouts accelerated.
- **Coverage:** App control moved from being SCADA-only to broad workstation coverage in pilot organizations.
- **User experience:** Blocked legitimate updates and user complaints decreased measurably.
- **Security posture:** The organization retained a strict default-deny stance while enabling legitimate change through auditable trust signals.

Lessons learned

- **Design for admin ergonomics:** Security controls that are costly to operate will not be adopted; operational simplicity is as important as technical correctness.
- **Multiple trust signals win:** Combining whitelisting with publisher, updater, path, and user trust balances security and usability.
- **Auditability is essential:** Every trust decision must be logged and revocable to maintain control and compliance.

- **Start small, iterate fast:** Pilots with clear telemetry accelerate safe scaling and build stakeholder confidence.

Conclusion

The Trust Engine reframes application control from brittle, hash-based enforcement to a flexible, auditable model where every executable must have a verifiable reason to run. That shift reduced administrative overhead, removed update friction, and made app control practical for high-change environments, delivering a 400% increase in adoption while preserving a secure default-deny posture.