AIRLOCK DIGITAL

TECHNICAL GUIDE

How Modern Application Control Works

Policy models, enforcement mechanisms, and operational workflows for controlling software execution

Why Controlling Software Execution Still Matters

Enterprise endpoints execute a continuous stream of software, including applications, scripts, installers, services, and transient processes. In modern environments, this execution surface is highly dynamic. Applications update frequently, new tools are introduced regularly, and users operate across a wide range of workflows.

At the same time, attackers continue to rely on unauthorized or unexpected software execution as a primary method of compromise. Whether through malware, fileless techniques, or living off the land (LOL) binaries, execution remains the moment where intent becomes action. Once software begins executing, attackers can establish persistence, escalate privileges, and move laterally.

Most organizations have responded by investing in detection and response technologies, such as endpoint detection and response (EDR). These tools provide visibility into suspicious behavior, enable rapid investigation and containment, and significantly improve visibility into endpoint activity. However, they are inherently reactive, identifying suspicious behavior after execution begins. In practice, this means organizations often determine whether software is trusted only after it has already run.

Application control enforces a simple but powerful principle.

Only trusted software should be allowed to execute.

Detection does not prevent execution. It answers the question:

What just happened?

Application control addresses a different, more foundational question:

Should this software be allowed to run?

Despite its effectiveness, many organizations historically struggled to deploy application control successfully. Earlier approaches treated allowlisting as a technical problem, focusing on blocking and allowing files, rather than an operational one.

Modern application control recognizes that execution control is not just enforcement. It treats application control as an ongoing operational capability centered on defining trust. Modern application control approaches provide the visibility, policy management, and workflow integration needed to define and maintain trusted execution as an ongoing operational discipline.

Traditional Allowlisting Models

Traditional allowlisting solutions rely on static policies that explicitly define which applications are permitted to execute. These policies are typically built using rules, such as:

- File hashes (exact binary matches)
- File paths (approved directories)
- Digital signatures (trusted publishers)

When fully enforced, this model provides strong preventive control by blocking any software not explicitly approved. In practice, however, maintaining these policies in enterprise environments is difficult.

Several reasons why these static models break down include:

- **Software is constantly changing:** Modern applications update frequently, and even small updates can invalidate rules.
- **Execution context is not static:** An application may behave differently depending on user roles, system configurations, or runtime conditions.
- **Policy ownership is often misaligned:** Security teams often define policies, but IT operations and application owners understand actual usage patterns.
- **Exception handling is continuous:** New applications, updates, and scripts are introduced daily, but static models do not account for this ongoing change.
- **Operational friction accumulates:** As policies grow more complex, managing them becomes increasingly difficult, leading to delays, errors, and lower adoption.

Another challenge is that traditional allowlisting models assume application control can be managed as a centralized security function. However, the understanding of what should be allowed to run is distributed across the organization.

As a result, allowlisting often becomes disconnected from operational workflows. Policies are either too restrictive and risk disruption or too permissive, which reduces effectiveness. Over time, many organizations experience policy drift as rules become outdated or are inconsistently enforced.

The core issue is not the effectiveness of allowlisting as a strategy. It is the difficulty of sustaining it as an operational process in dynamic environments.

Native Operating System Enforcement Tools

Modern operating systems include built-in enforcement mechanisms designed to support application control. In Windows environments, this includes Microsoft AppLocker and Windows Defender Application Control (WDAC). These technologies provide the underlying enforcement engine that determines whether software is allowed to execute.

At a technical level, these controls operate by intercepting process execution and evaluating the file against defined policies. When configured correctly, these mechanisms provide strong and reliable control over what can run on an endpoint.

These technologies operate at a low level in the system and provide reliable enforcement. However, native tools do not address the broader operational challenge of defining and maintaining those policies.

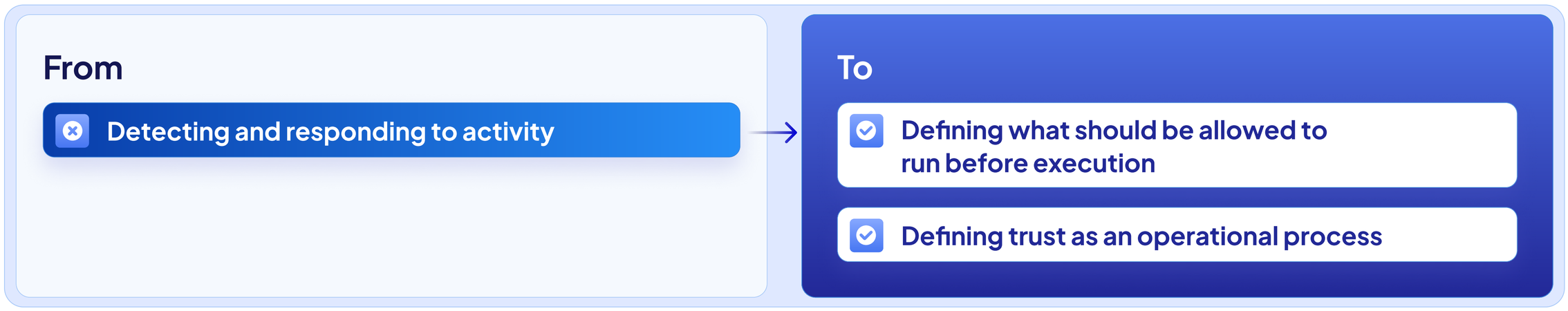
Among the limitations of native operating system enforcement tools are:

- **No visibility before enforcement:** Organizations often lack insight into what software is running before policies are applied.
- **Policy creation complexity:** Defining policies, especially in WDAC, requires deep expertise and careful configuration.
- **Lack of safe testing workflows:** Changes to policies can have an immediate impact, but there is limited ability to simulate outcomes.
- **Limited operational scalability:** Managing policies across large environments requires coordination that native tools do not provide.

Native operating system enforcement tools provide strong application control, but they do not address how policies are defined, tested, and maintained over time. Without visibility and workflow integration, organizations struggle to apply these controls consistently at scale.

The Modern Application Control Model

Modern application control represents a shift from static rule enforcement to an operational model for defining and maintaining trusted software execution. At its core, application control is not a technology problem. It is a process problem that must align with how organizations manage software, users, and change. Rather than treating allowlisting as a one-time configuration, this approach recognizes that execution control must evolve continuously alongside applications, users, and business workflows.



At its core, this modern application control model moves away from blocking unknown files to defining what is allowed to run—intentionally and contextually. Modern application control focuses on defining trust, not just enforcing rules. This requires answering:

- What software is trusted?
- Who is allowed to run it?
- Under what conditions?
- How should trust evolve?

These cannot be one-time decisions. They should be an ongoing process. It must be built into operational processes with trust decisions aligned with how software is actually used.

Core Capabilities of Modern Application Control

Application control must move beyond being a purely technical function

It must combine:

- Visibility into execution behavior
- Flexible policy models
- Incremental policy development
- Structured workflows for policy management
- Operational governance
- Multi-persona operational model

This combination transforms application control into a sustainable operational function, enabling organizations to define what software is allowed to run with precision, consistency, and minimal disruption.

Visibility into Execution Behavior

Visibility is foundational to modern application control models. To gain effective visibility, organizations must understand what is executing across their environment, **including:**

- Applications and binaries
- Scripts and interpreters
- Parent-child process relationships

Visibility into what is actually running in an environment, as well as execution frequency and patterns, provides the context needed to ensure policies are built on real execution behavior rather than assumptions. Without this visibility, enforcement decisions are incomplete and prone to disruption.

Flexible Policy Models

Modern application control solutions support flexible policy models that reflect real-world usage and execution patterns, including the following:

- **Binary-level controls:** Precise control using hashes or signatures for known software
- **Publisher trust models:** Allows software signed by trusted vendors to execute without requiring frequent policy updates
- **Installer-based trust:** Allowing software installed through approved deployment mechanisms
- **Context-aware policies:** Defining rules based on user roles, system types, or environmental conditions
- **Process relationship controls:** Controlling how processes spawn and interact, preventing misuse of legitimate tools

Blending these controls allows organizations to define policies that reflect how software behaves in practice. This enables more precise enforcement by governing how applications execute and interact, reducing the risk of legitimate tools being misused.

Incremental Policy Development

Modern application control avoids the need to define a complete allowlist upfront. Instead of attempting to define a complete allowlist upfront, **organizations should:**

- Observe execution activity
- Identify legitimate software
- Approve known software and trusted applications

Trusted policies can be expanded, over time, based on observed execution activity and identified legitimate software. This enables a controlled transition toward stricter enforcement models, including deny by default, without introducing operational risk.

Structured Workflows for Policy Management

Modern application control introduces structured workflows for policy management.
Exception handling, policy updates, and change management are integrated into existing operational processes. This ensures that execution control remains aligned with how software is introduced, updated, and used across the organization.

Operational Governance

Application control is transformed into an operational discipline with continuous governance-related processes, **including:**

- Policy updates
- Exception management
- Application lifecycle tracking
- Ongoing refinement

Integrating governance into operational workflows helps organizations own trust decisions, defining what is allowed to run based on their environment and risk tolerance.

Multi-Persona Operational Model

Traditional approaches assumed centralized control. Effective application control requires determining what should run based on input from multiple personas, and defining roles across teams, including security, IT operations, service desk, and application owners.

Team	Primary Function	Role
Security	Risk and governance	Define risk boundaries and governance requirements.
IT operations	System management	Manage systems and understand software usage.
Service desk	Exceptions and requests	Handle user requests and exceptions.
Application owners	Business requirements	Understand business requirements and dependencies.

Application control must align with how these teams already operate. **For example:**

- Service desks handle user software requests
- IT teams validate operational requirements
- Security teams define policy boundaries

Modern application control solutions support this by enabling role-based workflows that allow different teams to contribute while maintaining centralized oversight, bringing these personas together into a coordinated process.

Legacy vs. Modern Application Control

Legacy Approach	Modern Approach
Static allowlists	Incremental policy development
Manual updates	Workflow-driven policy management
Limited visibility	Full execution visibility
High operational overhead	Operational governance

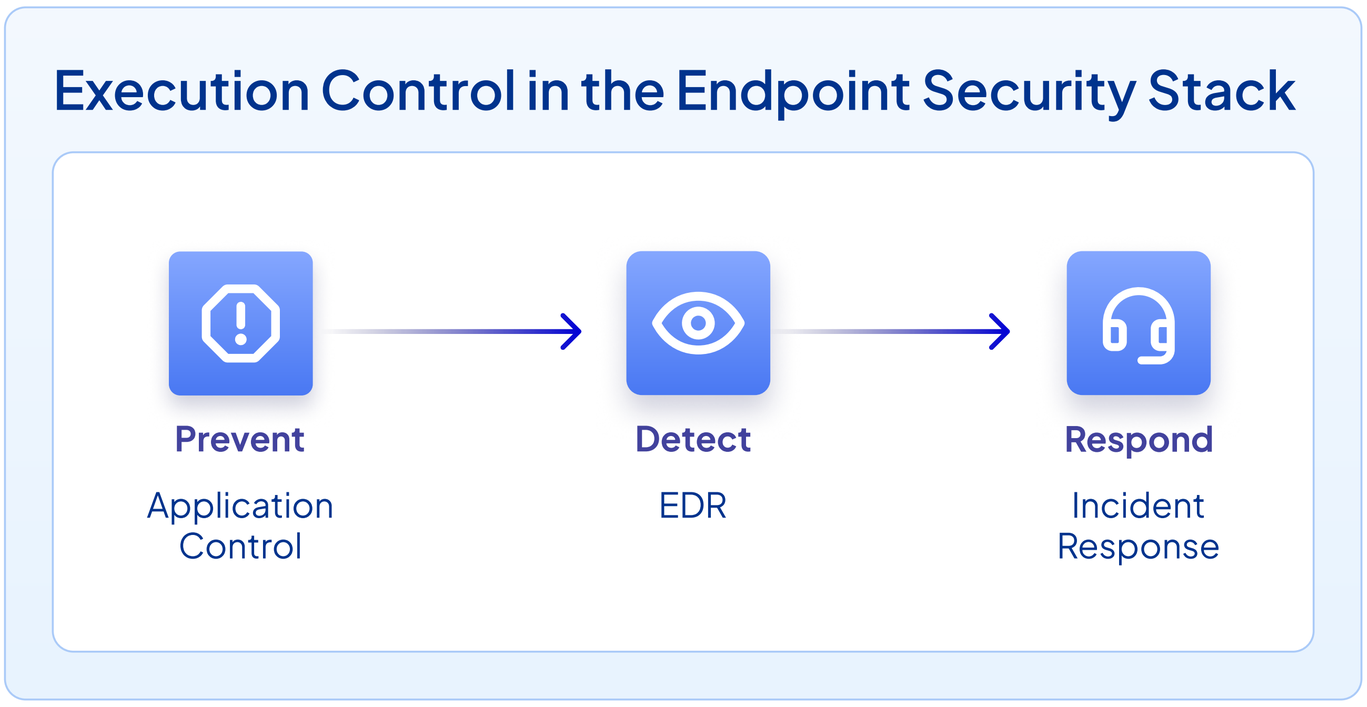
Where Application Control Fits in Endpoint Security

Application control functions as a preventative control within a layered endpoint security architecture. While technologies such as endpoint detection and response (EDR) provide visibility into suspicious activity and enable investigation after execution begins, application control operates earlier in the attack chain by determining whether software is allowed to run at all.

This distinction is important in practice. Detection tools are designed to identify and respond to behavior, which can generate large volumes of alerts—many of which require investigation to determine risk. Application control reduces this burden by limiting the number of unknown or untrusted processes that can execute in the first place.

By enforcing trusted execution, application control helps create a more predictable operating environment. This improves the effectiveness of downstream controls by reducing noise and allowing detection technologies to focus on genuinely anomalous or high-risk activity.

Application control does not replace detection and response. Instead, it complements these capabilities by addressing a different layer of the problem. Detection answers what happened after execution, while application control determines what is allowed to happen before it begins.



This shift reflects a move from reactive security to intentional execution control, where organizations define acceptable behavior before execution occurs.

Enforcement Mechanics: How Modern Application Control Works

Application control operates at the point of process execution. At its core, application control makes a decision before execution occurs. When a process is initiated:

1. **The system intercepts the execution request.**
2. **The binary is evaluated against policy.**
3. **Contextual factors are applied (e.g., user, system, environment).**
4. **A decision is made to allow or block execution.**

This evaluation occurs prior to process execution, ensuring that unauthorized software is blocked before it can run and preventing untrusted code from establishing a foothold on the endpoint.

Application control operates pre-execution, preventing processes from starting.
Some advanced controls also monitor runtime behavior, including:

- **Process lineage (parent-child relationships)**
  Evaluates how processes interact to prevent the misuse of legitimate tools.
  For example:
  - A trusted application launching another trusted application is allowed.
  - A trusted application launching an unknown script is evaluated or blocked.

- **Real-time policy evaluation**
  Allows policies to be updated dynamically **to enable:**
  - Immediate response to new software requests
  - Rapid policy adjustments
  - Reduced operational delays

This allows organizations to control execution in context, not just block binaries. By including context, modern application control platforms ensure that trust decisions reflect how software behaves in real environments and that enforcement decisions are both accurate and adaptable.

Operational Workflows for Sustaining Application Control

Application control must align with how organizations already manage software, endpoints, and change. Sustainable execution control is not achieved through a one-time policy deployment. It requires repeatable workflows that integrate with IT operations, service management, and security governance. The following are the five key steps to sustaining application control.

1 Baseline and Discovery

Organizations should begin by observing execution activity across endpoints to establish a baseline of which software is actually in use. This includes identifying applications, scripts, and supporting processes, as well as how they are invoked and by whom. The goal is not just inventory but also understanding context, such as frequency, user groups, and dependencies. This visibility allows teams to distinguish between expected operational software and unknown or unnecessary execution.

2 Trust Definition

With a baseline established, organizations should define trusted software based on business need, risk, and usage patterns. This process typically involves collaboration between IT operations, application owners, and security teams. Trust decisions should incorporate factors such as publisher reputation, installation method, system role, and user context. These decisions are owned by the organization, reflecting its environment rather than relying on generic trust models.

3 Exception Management

New software requests, updates, and edge cases should be processed through structured workflows. Service desks often act as the intake point, with IT validating operational requirements and security assessing risk. Effective workflows ensure that exceptions are evaluated consistently, approved quickly where appropriate, and recorded for auditability. This prevents ad hoc policy changes and reduces friction for end users.

4 Controlled Enforcement

Enforcement should be introduced incrementally to avoid disruption. Teams typically begin by monitoring or auditing to understand the impact of policies before blocking execution. Enforcement is then applied in stages, by user group, system type, or environment. This allows teams to validate outcomes and adjust policies as needed. Following this gradual progression supports a predictable path toward stricter models such as deny by default.

5 Continuous Maintenance

Application environments evolve continuously, requiring policies to be updated and refined as they evolve. This includes following approved workflows for software updates, onboarding new applications, and removing outdated or unused entries. Ongoing coordination between security, IT operations, and service teams ensures that workflows and associated policies remain aligned with real-world usage. Over time, this transforms application control into an operational discipline that consistently governs what runs without disrupting the organization's operations.

Operationalizing Trusted Execution

Application control remains one of the most effective ways to reduce endpoint attack surfaces, but its success depends on its operational sustainability. Traditional allowlisting approaches struggled not because the strategy was flawed, but because organizations lacked the visibility and workflows needed to maintain policies in environments where software and usage constantly change.

Modern application control addresses this by turning execution control into an ongoing operational process. Teams define trust based on what actually runs in their environment, apply policies incrementally, and integrate enforcement into existing IT and security workflows. This enables organizations to move toward models such as deny by default in a controlled and predictable way, rather than through disruptive, one-time changes.

Airlock Digital shows how modern application control can be operationalized in practice. By providing execution visibility, flexible policy management, and workflow-driven governance, organizations can define and maintain trusted execution over time, reducing exposure while maintaining operational continuity and control over what runs.

AIRLOCK DIGITAL

Airlock Digital is an application control company that helps organizations define what they trust and control what runs across their environments. Founded in Australia in 2013, Airlock Digital works with enterprises worldwide to reduce exposure to unauthorized software and strengthen control over software execution in complex environments. By helping organizations move from implicit execution to organization-defined trust, Airlock Digital supports a more intentional and sustainable approach to prevention. This approach helps security and IT teams improve control, reduce uncertainty around what is allowed to run, and maintain operational stability while strengthening existing security investments.

Connect with Airlock Digital

www.airlockdigital.com