



Security + Stability + Strategy = Success

## How Can You Achieve Zero Trust Endpoint Protection?

The point of applying zero trust within the endpoint is to reap better protection for less effort. Alternatives are ineffective and labor intensive because they monitor and investigate vast, diverse volumes of detection and indicator data from multiple perspectives at multiple stages of malware attacks: before and after compromise. They parse infinite possibilities, requiring more tools, more personnel, and more skills every year. Those who think machine learning will help them scale are finding that the single most pervasive characteristic in enterprise IT — CHANGE — is also machine learning’s greatest adversary. A far different approach is needed.

### **A proven Approach:**

Rather than trying to scale to parse more, this Zero Trust Endpoint Protection takes the opposite approach: drastically reducing what needs to be monitored and analyzed. It does this by avoiding the predicament of telling “good” from “bad” and “normal” from “abnormal” by instead blocking those actions malware needs to take to execute. This replaces analyzing an infinite amount of data with suppressing hundreds of actions within an endpoint that years of industry research have revealed are necessary for adversaries to attain their goals. With this approach, malware recognition is not required. Alternatives only succeed when they can recognize **every** piece of malware.

Malware’s actions are performed by processes, which spring from applications, utilities, and untrustworthy files. Zero Trust expects applications and utilities to go rogue at any moment. It uses containment, isolation, default-deny, and other controls to disrupt malware’s intended actions. And it doesn’t make statistical guesses. If it cannot deterministically block, then it restrains. SecOps is spared from the tug of war between false positives and false negatives.







Failed conformance controls such as whitelisting, HIPS, and sandboxing require too much endpoint state information that needs to be revised following changes such as application updates/patches. This endpoint Zero Trust is based on patented higher abstractions that simplify policy formulation and automatically adapt to lifecycle changes. For example, app containment begins with its parent executable and automatically extends to any resulting process from the app’s operation. This means very little state information is required for policy formulation, and updates/patches do not necessitate policy updates. Further, it accounts for the unanticipated.

Over 90% of enforced policies are defined by default. Agents typically run many months without policy updates — some have run for years. Containment is enforced uniformly to all at-risk apps, avoiding the app-specific policy dilemmas of alternatives.

Customers praise this real-time protection effectiveness and its near set-and forget operations. Zero Trust Endpoint Protection defeats malware without having to detect it, resulting in better protection and fewer operations. Further, other cyber defense layers see substantially lower alert volumes because malware attacks are stopped at endpoints in real time.

## The Zero Trust Endpoint Protection Framework

### WHAT IS POLICY-BASED ZERO TRUST FRAMEWORK?

-  **Contain** unacceptable actions from high-risk applications and utilities
-  **Isolate** access and/or alteration of resources
-  **Deny** launch of untrustworthy executables, scripts, and remote code
-  **Reduce** exposure from unnecessary utilities and capabilities
-  **Permit** use of capabilities suppressed by “deny” and “reduce”
-  **Demote** processes created in specific ways, rendering them harmless

Use Case	How Zero Trust Mitigates Risks and Accommodates Legitimate Usage
Unpatched app or zero-day exploit	Does not allow an app or any process it spawns to install malware or steal/alter the memory of other app/OS processes. This alleviates patch/vulnerability management pressure. Containing an app is as simple as adding a song to a playlist.
Drive-by download	Scripts and executables are not allowed to launch unless proven trustworthy via validated digital signature or other means; those allowed to launch are not allowed to do harmful actions.
Server with mission-critical app has mysterious, malicious process running	Any malware that somehow gets onto a server cannot read/write the memory, directories, executables, or data files of the “isolated” mission-critical app. IT/Sec-Ops can usually safely run the app until a maintenance window.
Pass-the-hash/ticket attacks	Blocks credential thefts by granting access to trustworthy processes only. No IT/Sec-Ops actions are required; eliminates alerts that other tools would otherwise make.
Non-malware attacks	Prevents unauthorized actions by built-in tools yet allows limited use by end-users and full-use by IT/Sec-Ops. This requires fewer than a dozen deployment-specific policy rules that rarely require adjustment later.
Code injection attacks	Blocks clearly untrustworthy app process changes and ensures the app’s processes cannot do harmful actions in case they ever do run malicious code. Spares IT/Sec-Ops from the false-positive/negative quagmires of behavior analytics and other tools.
Remote code execution attacks from other endpoints	These built-in capabilities (e.g., Remote PowerShell, PsExec-like, SSH/shell, etc.) are locked/unlocked to ensure only IT/Sec-Ops can use them on demand, even if adversaries somehow steal elevated privilege credentials.