

Putting NSA's OT Zero Trust Guidance into Practice

A Mission-Focused Path to Containment and Trustworthy Recovery

Dr. Ron Ross

CEO, RONROSSECURE, LLC
ISTS FELLOW, DARTMOUTH COLLEGE

Abstract

The National Security Agency's guidance for hardening critical operational technology (OT) environments with zero trust principles provides a practical starting point for organizations protecting legacy systems. Its recommendations recognize the constraints of aging equipment, operational continuity, and physical safety. This paper offers an implementation companion that connects selected NSA activities to mission outcomes using NIST SP 800-160, Volumes 1 and 2, and Engineering for Compromise. A hypothetical water utility illustrates how a team can define unacceptable losses, restrict consequential authority, examine a credible compromise, establish protective operation and recovery, and gather evidence safely. The proposed pilot produces a small set of reviewable engineering artifacts rather than a claim that an entire environment has become resilient. Near-term changes are distinguished from longer-term acquisition objectives. The aim is to help organizations demonstrate what essential service and protection remain dependable when selected system elements become untrustworthy, while making unsupported scenarios, operational tradeoffs, and residual risk explicit.

1. Building on a Practical Starting Point

NSA's October 2026 Cybersecurity Information Sheet, *Hardening Department of War (DoW) Critical Operational Technology (OT) Environments with Zero Trust (ZT) Principles*, identifies 26 high-impact activities selected from the broader OT zero trust framework [1]. The guidance offers compensating protections for legacy equipment that cannot readily support modern security agents, authentication, or updates. Its emphasis on restricted remote access, controller hardening, segmentation, inventories, and safe assessment gives operators a manageable starting point.

That contribution is important. Operators must protect equipment that supports physical processes, often with limited maintenance windows, vendor dependencies, and little tolerance for interruption. A recommendation that overlooks those conditions may be technically attractive and operationally unusable. NSA addresses them directly, including the need to evaluate security changes for their effects on safety, availability, and process reliability.

Every activity list, however well chosen, shares one natural limit: controls implemented are not the same as outcomes demonstrated. A list can tell operators what to do; it cannot by itself show what essential service and protection remain when something goes wrong in a specific environment. Closing that gap is work each mission team must do for its own systems.

Organizations can therefore build on these recommendations by answering a practical question: after the selected measures are implemented, what can the team demonstrate about protection of the essential service when something goes wrong? The answer connects implementation to mission outcomes. It helps decision makers understand which consequences have been constrained, which system elements remain necessary to protection, and what still requires investment or risk acceptance.

NIST SP 800-160, Volume 1 provides the systems security engineering foundation for that work [2]. Volume 2 supplies cyber resiliency concepts for continuing essential functions, limiting damage, and

restoring trustworthy resources [3]. *Engineering for Compromise* applies those foundations to selected losses of trust and the consequences available through compromised system elements [4]. The method proposed here complements NSA's guidance; it is an author-developed implementation approach, not an additional NSA requirement or a prescribed NIST sequence.

Begin with one essential service and a protection claim the team can demonstrate.

2. Establish a Small Mission Boundary

Begin with one function whose loss has a clear operational consequence. A pumping function, backup generation service, or building control function is a more manageable pilot than an entire installation. Include the resources that enable the function: controllers, sensors, communications, power, vendor access, engineering workstations, administration, and recovery. The initial boundary may expand when dependency analysis reveals a resource essential to the claim.

Assign an accountable mission owner and a lead architect or engineer. In a small organization, the lead may be an existing engineering manager; creating a new position is not necessary. Involve operators, safety personnel, security staff, and the vendor or integrator early. The mission owner decides service priorities and accepts residual risk within their authority. The engineering lead maintains the integrated dependency picture and coordinates changes that cross team boundaries.

The first deliverable is a one-page statement of the function, its operating modes, unacceptable losses, and minimum acceptable service. Specify duration, scope, and operating conditions where they matter. A statement such as "maintain water service" is too broad to guide a test. Define which demand must be met, for how long, with what pressure and quality constraints, and under which available power and storage conditions. Operators and safety engineers must establish the values from site-specific process analysis.

Do not invent a service threshold to make the pilot convenient. If the organization cannot yet define acceptable degraded operation, record that as an engineering task. A defined protective shutdown may be appropriate for some conditions, while a local control mode may preserve service for others. Choosing between them requires analysis of the consequences of continuing and stopping.

3. A Worked Example: Protecting a Water Pumping Function

Consider a hypothetical utility with a supervisory workstation, remotely maintainable pump controllers, operational telemetry, and local protective controls. The mission is to deliver required water supply while avoiding hazardous pressure and equipment damage. This example illustrates decisions and artifacts; it reports no deployed design or measured performance.

For the pilot, the utility identifies three unacceptable outcomes:

- Hazardous pressure caused by commands or configuration changes that defeat the approved operating envelope.
- Loss of required minimum supply beyond the mission owner's approved duration under specified conditions.
- Return to service with controller logic or protective settings whose suitability and integrity have not been established.

The team sketches the paths by which the workstation, vendor account, controller maintenance interface, or restoration process can influence those outcomes. It records both normal command paths and exceptional paths used for firmware updates, diagnostics, emergency access, and recovery. An inventory supplies the system elements; the dependency diagram shows how their capabilities combine.

The team then defines constraints. Supervisory commands must remain within the approved process envelope. Ordinary maintenance access must not silently change protective limits. Essential local control must remain available for the defined period after remote access is isolated, if the process supports that mode. Restoration must meet acceptance criteria before consequential authority is reenabled.

These constraints do not assume the existing protective controls are independent or adequate. The team must determine where they execute, who can modify them, and whether they share firmware, sensors, power, administration, or update paths with the equipment they protect. A limit implemented inside a compromised controller may constrain a hostile workstation while providing no protection against compromise of that controller itself. The claim must identify that difference.

4. Apply NSA Activities to the Mission Constraints

Select NSA activities that address the identified paths, using the guidance's activity identifiers to preserve traceability [1]. The following applications are an illustrative selection for a single pilot rather than an exhaustive implementation checklist or a complete mapping of all 26 activities.

- **Remote access:** Remove direct Internet exposure where present. Use an approved mediated access path, phishing-resistant multifactor authentication where supported, explicit vendor approval, time limits, and termination after maintenance. Relevant activities include 1.2.2, 1.3.1, 1.4.1, 1.5.1 [1].
- **Authority:** Inventory accounts, remove unnecessary privileges, and distinguish monitoring, operational control, and maintenance capabilities. Activities 1.1.1, 1.2.1, and 1.7.1 support this work. Determine whether a permitted maintenance operation can indirectly replace logic that later runs with greater authority [1].
- **Controller functionality:** Review enabled logic and services against operational need. Disable unused functions through approved, vendor-supported procedures and verify the resulting process behavior. Activities 2.6.2 and 3.3.1 support controlled configuration and application control [1].
- **Communication boundaries:** Permit required paths and protocols and constrain unnecessary interaction using segmentation and connection policy. Activities 2.2.1, 5.3.1, and 5.4.1 provide relevant implementation steps. Examine administrative paths as well as process traffic [1].
- **Visibility:** Maintain the asset and code inventories, operational baseline, protected logs, and telemetry needed to detect departures from the constraints. Activities 2.1.1, 3.1.1, and 7.1.2 contribute evidence about deployed state and behavior [1].

SP 800-160, Volume 1's *Security Design Order of Precedence* helps order the choices: eliminate unnecessary loss paths where feasible, reduce remaining loss potential through design, then provide protective mechanisms, visibility, and supporting procedures [2]. Table 1 shows the order of precedence with examples drawn from the utility pilot. Removing an unneeded remote write function, for example, avoids having to continually police its use. Necessary remote maintenance still requires dependable mediation and operational coordination.

Table 1. Security Design Order of Precedence applied to the utility pilot (most effective first)

	ALTERNATIVE (SP 800-160 VOL. 1)	EXAMPLE FROM THE UTILITY PILOT
1	Eliminate the potential for loss through design selection.	Remove direct Internet exposure and unneeded remote write functions, so those paths no longer exist.
2	Reduce the potential for loss through design alteration.	Separate maintenance authority overprotective settings from the authority to issue operational commands.
3	Incorporate engineered features or devices.	Mediate remaining remote access through an approved path with phishing-resistant multifactor authentication, time limits, and termination after maintenance.
4	Provide visibility and feedback.	Maintain protected logs, the operational baseline, and telemetry that reveal departures from the constraints.
5	Incorporate signage, procedures, training, and equipment.	Approved maintenance procedures and operator training for local and degraded operation, supporting rather than replacing the measures above.

The activity identifier is useful for implementation tracking. The mission constraint explains why the activity matters, and the acceptance criterion states what must be demonstrated. Keep all three together in the engineering record.

5. Examine One Credible Compromise

Start with a scenario the team can analyze: the engineering workstation is controlled by a hostile actor who can exercise its existing privileges. Identify the compromise level and assumptions. Application compromise, operating-system compromise, and controller firmware compromise are different scenarios and may require different barriers. The pilot can address one first without implying that the others have been resolved.

Map what the hostile workstation can read, alter, invoke, delegate, exhaust, or disable. Include its ability to send valid commands, change controller logic, reuse credentials, reach maintenance interfaces, consume shared resources, and corrupt restoration files. Authentication may continue to succeed. The engineering question is what consequential effects remain possible through permitted capabilities.

For each path to an unacceptable outcome, decide whether to remove authority, narrow it, add independent enforcement, or identify an unsupported scenario for escalation. In the utility example, the workstation might retain permission to propose or issue bounded operational commands while losing ordinary access to protective settings. Changes to those settings could require a separate maintenance path and approval authority. Whether this is feasible depends on the controller and the operational workflow.

Independence requires more than a second account or another network segment. Determine whether one administrator, management service, update package, or identity compromise can defeat both the command boundary and the protective limit. Where the architecture cannot support the required independence, reduce the scope of the claim and present the remaining consequence to the mission owner.

A practical test for independence

Ask these questions of each protective mechanism claimed as independent of the system element assumed to be hostile:

- Can a single compromised identity, credential store, administrator, or management service alter both the operational command path and the protective limit?
- Are the protective settings held where the untrusted system element cannot write to them, even with all of its current privileges?
- Does the protective mechanism continue to operate if the supervisory network, identity service, or update channel is unavailable or behaves maliciously?
- Has a representative test shown that the barrier cannot be bypassed through diagnostics, vendor connections, emergency accounts, or firmware update paths?

If the answer to the first question is yes, or the answer to any of the remaining three questions is no, the independence claim is not yet substantiated. Record the relevant vulnerability, dependency, or evidence gap and present it to the mission owner for explicit disposition.

A valid identity does not establish that every permitted action is safe for the mission.

The pilot's bounded claim might be: within the defined operating conditions, compromise of the supervisory workstation cannot change protective limits through the identified interfaces, and isolation of that workstation preserves the defined local service for the approved duration. State which controller, protective mechanism, credentials, and supporting resources must remain trustworthy. This is a claim to test, not a conclusion reached by drawing the diagram.

6. Check the Boundary and Its Supporting Dependencies

For each protection claim, identify the enforcing system element and its dependencies. SP 800-160, Volume 1 specifies four essential design criteria for protective mechanisms: (1) non-bypassable, (2) evaluable, (3) always invoked, and (4) tamper-proof [2]. Applying those criteria to OT produces practical questions.

Table 2. Essential design criteria applied to OT protective mechanisms

DESIGN CRITERION	VERIFICATION QUESTION	OT FOCUS (EXAMPLES)
Non-bypassable	Can diagnostics, a vendor connection, an emergency account, or a restoration operation bypass the constraint?	Maintenance ports, secondary communication paths, and local manual overrides.
Evaluatable	Is the enforcement function understandable enough to assess its claimed behavior? What design information, configuration evidence, analysis, and tests are available?	Controller logic, gateway policy rules, and documented operating states.
Always invoked	Does the constraint remain effective during maintenance, failover, degraded operation, and recovery within the claim's scope?	Protective interlocks during local, emergency, and startup modes.
Tamper-proof	Can the assumed hostile element change the enforcement policy, its software, or the data on which its decision depends?	Where protective settings are stored and who can write them; for example, read-only storage or a physical key switch where the equipment supports it.

Also examine failure of the protection function itself [2]. Loss of an identity service may prevent new maintenance sessions without requiring local control to stop. Loss of reliable telemetry may require a different protective response. A gateway that fails open could defeat the very constraint that it was introduced to enforce. The appropriate behavior must follow the mission and process analysis.

Document unresolved dependencies explicitly. For instance, shared power does not automatically invalidate a cyber containment claim, but it does limit a claim about service continuity during power loss. A shared deployment service may directly invalidate independence against malicious updates. The effect depends on the *adversity* being claimed as tolerated.

Acquire evidence commensurate with the consequence of failure. A relevant Common Criteria evaluation can contribute evidence for an enforcement component, subject to its evaluated version, configuration, scope, and assumptions [5]. Vendor design information, tailored assessment, and independent testing can provide other evidence. Component evidence must be connected to the deployed composition; certification alone does not establish the utility's mission-level claim.

7. Define Protective Operation and Trustworthy Recovery

Before isolating a workstation or disabling vendor access, establish what remains in control of the process. Define operator responsibilities, necessary indications, essential resource availability, and the transition into local or degraded operation. Confirm whether operators have the staffing, training, access, and equipment needed to sustain that mode for the claimed interval.

Do not prescribe automatic shutdown or automatic continuation for every anomaly. Both can create unacceptable consequences. Safety and operations personnel should define the response to specific loss-of-trust conditions and verify it against the process. Emergency procedures must preserve the necessary constraints rather than quietly reintroduce unrestricted access.

Recovery should restore the basis for justified trust, including appropriate controller logic, protective settings, credentials, authoritative data, and operational state [3; 4]. Protect restoration artifacts and recovery authority from the primary system's identified compromise paths. A backup stored in another location may remain vulnerable if the compromised workstation can alter it or use the same credentials to administer it.

Record restoration provenance and acceptance criteria. A valid signature helps establish origin and integrity, but does not establish that the approved logic is correct for the present process. The team must verify the selected configuration's suitability, test required protective behavior, and determine when remote command or maintenance authority can safely be restored.

The deliverable is a short fallback and recovery procedure with named decision authority, required resources, verification steps, and criteria for returning to service. Exercise the procedure under the same assumptions used in the claim. Restoration time is useful only when the restored capability also meets its acceptance conditions.

Trustworthy recovery restores the conditions for justified trust and essential service.

8. Gather Evidence Without Endangering Operations

NSA appropriately prioritizes passive assessment and carefully coordinated active testing for sensitive OT [1]. Configuration review, logs, and passive traffic analysis can establish deployed settings and

observed communication paths. Use them first to validate the inventory and identify discrepancies that affect the protection claims.

Some claims require additional evidence. Passive observation of ordinary traffic does not show that a prohibited maintenance path is blocked, that a gateway cannot be bypassed, or that fallback survives loss of a dependency. Exercise these conditions in a representative laboratory, spare controller setup, or other approved test environment where feasible. Document the differences from the operational installation and explain which conclusions can transfer.

Plan tests with operators, safety personnel, and the vendor. Specify prerequisites, authorized actions, expected results, stop conditions, restoration arrangements, and evidence collection. An approved maintenance window is necessary for some production work but is not by itself a sufficient safety justification. If representative testing is unavailable, narrow the claim, use other justified evidence, and record the limitation.

The utility's initial evidence plan can include the following checks:

- Attempt a prohibited protective-setting change from the representative compromised-workstation context; verify rejection at the claimed boundary and preservation of the setting.
- Exercise the identified alternate maintenance paths; establish that required mediation remains effective or record a bypass for correction.
- Remove the supervisory connection in the approved test setup; demonstrate the defined local control behavior under representative demand and resource conditions.
- Make the access service unavailable; verify the intended response for new and existing sessions and essential process control.
- Restore an approved controller configuration using protected artifacts; verify process suitability and protective behavior before restoring consequential authority.

For each check, record the configuration, assumptions, expected result, observed result, and responsible reviewer. A successful test supports the claim within its scope; it does not establish protection against every attack. A failed test is useful evidence for changing the design or reducing the claim.

9. Run a Focused Pilot with Reviewable Outputs

A proposed 90-day pilot can organize the work. Vendor availability, maintenance windows, process complexity, and procurement often extend it, so the schedule is a planning aid, not a deadline for declaring success.

During the first 30 days, define the mission boundary, unacceptable losses, operating assumptions, and minimum acceptable service. Review the dependency diagram and authority map with operations and safety personnel. Select one compromise scenario and a small number of relevant NSA activities. Approve the evidence plan before making consequential changes.

During days 31 through 60, implement the highest-priority feasible changes. These may include removing unnecessary exposure, narrowing an account, separating maintenance authority, or protecting restoration artifacts. Evaluate each change for latency, reliability, vendor support, operator workload, and effects on other essential functions. Use the test environment to resolve uncertainties before deployment.

During days 61 through 90, exercise the targeted constraints, fallback, and recovery. Review failed checks and incomplete evidence. Present the demonstrated claim, remaining dependencies, implementation costs, and residual risk to the mission owner. If the evidence is insufficient, the pilot should conclude with an explicit next action rather than an unsupported resilience claim.

Keep the deliverables small and useful:

- A one-page mission and loss statement approved by the accountable owner.
- A dependency diagram and authority map that include maintenance and recovery paths.
- A record linking selected NSA activities to constraints and acceptance criteria.
- A fallback and recovery procedure with decision authority and return-to-service criteria.
- An evidence record identifying demonstrated claims, unsupported scenarios, and residual risk decisions.

Exit criteria should describe outcomes. Each targeted loss path is eliminated, constrained by a substantiated barrier, or presented for explicit risk disposition. Essential fallback and restoration meet their defined criteria, or the limitation is recorded with a funded or otherwise accountable next step. Completion of activity checkboxes remains useful progress information but should accompany these mission-based results.

What the mission owner receives

The pilot ends with a short presentation of what was demonstrated and what remains. For the utility example, it might read:

“Under the examined compromise, hostile control of the engineering workstation with its current privileges, testing showed that protective setpoints could not be changed through the supervisory interface, and that local control sustained the defined service for the tested interval. Two consequences remain possible. First, protective setpoints can still be changed through the vendor maintenance interface until the separate maintenance path is in place; consequence: hazardous pressure. Second, the primary restoration image can be altered because its store shares credentials with the workstation; consequence: delayed or untrustworthy recovery. Recommended disposition: schedule the separate maintenance path for the next maintenance window and accept the restoration risk for a stated period while a protected recovery store is implemented. Decision authority: mission owner.”

The example is hypothetical. Its value lies in the form: a bounded claim, the consequences that remain, a recommended disposition for each, and the person who decides.

10. Keep Costs and Organizational Obstacles Visible

Use existing diagrams, maintenance records, inventories, and configuration repositories where they are dependable. Consolidate them around the selected mission function instead of creating a parallel documentation program. Reserve operator and vendor time explicitly; an exercise without the personnel who understand the process is unlikely to provide credible evidence.

Measure analysis effort, engineering changes, testing, outages, equipment, and recurring upkeep. Record operational friction such as additional approvals or reduced maintenance flexibility. Removing unused capability may lower cost, while independent enforcement or recovery can require investment. The pilot should reveal those tradeoffs rather than promise a universal return on investment.

Common obstacles include a scope that expands faster than evidence can be produced, late involvement of operators or safety personnel, findings that cross organizational authority, and unavailable vendor information. Address them through a narrow initial claim, early participation, a named executive sponsor, and explicit acquisition or supplier requests. If an existing system element cannot be constrained adequately, consider withholding the consequential authority, an independently supported compensating barrier, replacement, or formal acceptance of the residual risk.

Maintain the record after the pilot. Changes to controller logic, accounts, network policy, firmware, vendor access, protective settings, or recovery artifacts can invalidate evidence. Define who reviews affected claims and which changes require retesting. Useful measures include demonstrated restoration time, time since the last fallback exercise, unresolved consequential paths, and the share of claims with current supporting evidence. These are proposed local measures, not NSA maturity requirements.

11. Use the Findings to Improve Future Acquisitions

Near-term compensating protections can reveal the capabilities that future systems should provide by design. Convert recurring difficulties into requirements for the next controller, gateway, management platform, or system upgrade. Procurement objectives should describe protection behavior and evidence rather than simply require a product labeled *zero trust*.

Useful objectives include separate authority for operational commands and protective configuration; documented consequential interfaces; independently protected enforcement appropriate to the identified losses; supported local operation under defined dependency failures; protected recovery resources; and supplier evidence sufficient to evaluate the relevant security properties. Request details about administrative, diagnostic, firmware-update, and restoration paths because those paths can defeat otherwise convincing separation.

Specify supplier deliverables and acceptance evidence proportionate to mission consequences. Suitable evidence may include design and interface descriptions, relevant evaluation results, configuration baselines, failure analyses, and agreed test results. Combine those with integration analysis and operational validation. SP 800-160, Volume 1 design principles of *Commensurate Trustworthiness*, *Substantiated Trustworthiness*, and *Compositional Trustworthiness* provide the basis for relating the required confidence to the system and its parts [2] [5].

A longer-term implication: AI tools

Looking beyond the near-term pilot, where AI tools are introduced for planning, diagnostics, or maintenance, allocate only the authority needed for their role. Generated requests and retrieved content should not acquire consequential authority simply because an approved agent processed them. Enforce relevant constraints outside the model and retain a tested essential-service mode without the agent. Changes to tools, retrieval, memory, and model deployment should reopen affected claims. These objectives follow the same loss-driven reasoning used for other powerful automated system components; the architecture must state what it depends on and what it can tolerate. They are acquisition and design objectives for future systems, outside the scope of the near-term pilot.

12. Demonstrate Progress One Mission Function at a Time

NSA's guidance creates an opportunity to make practical progress in difficult legacy environments. Its recommendations deserve implementation, operational tailoring, and continued investment. Systems security engineering can help organizations connect that work to the consequences their missions cannot tolerate.

A focused pilot makes the contribution tangible: choose one essential function, apply the relevant activities, examine a specific compromise, establish protective operation and recovery, and gather evidence safely. The result is a bounded protection claim with visible assumptions and limitations. That record helps leaders decide what to improve next and helps acquisition teams specify stronger systems over time.

Apply the guidance to a mission function and demonstrate the protection achieved.

The objective is steady improvement that operators can sustain. Each demonstrated constraint, dependable fallback, and trustworthy restoration step contributes to a more defensible account of mission protection under adversity.

This approach is a proposal to pilot, not yet a validated method. Organizations that try it on a single mission function are invited to share their artifacts, results, and lessons, including any unsupported scenarios and corrections, so that future editions can reflect operational experience.

Acknowledgments

The author acknowledges the Institute for Security, Technology, and Society (ISTS) at Dartmouth College for supporting the research and writing activities related to systems security engineering. The views expressed in this paper are those of the author and do not necessarily represent the views of Dartmouth College, the National Institute of Standards and Technology, or the National Security Agency. The worked example, pilot, and implementation artifacts are the author's engineering synthesis and do not imply NSA endorsement.

References

- [1] National Security Agency. *Hardening Department of War (DoW) Critical Operational Technology (OT) Environments with Zero Trust (ZT) Principles*. Cybersecurity Information Sheet, U/OO/6070841-26, PP-26-4325, October 2026, Version 1.0. Release and report: <https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/4621936/nsa-releases-zero-trust-guidance-for-protecting-operational-technology-systems/>
- [2] Ross, R., Winstead, M., and McEvilly, M. *Engineering Trustworthy Secure Systems*. NIST SP 800-160, Volume 1, Revision 1. November 2022. <https://doi.org/10.6028/NIST.SP.800-160v1r1>
- [3] Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., and McQuaid, R. *Developing Cyber-Resilient Systems: A Systems Security Engineering Approach*. NIST SP 800-160, Volume 2, Revision 1. December 2021. <https://doi.org/10.6028/NIST.SP.800-160v2r1>
- [4] Ross, R. "Engineering for Compromise: Designing Systems That Remain Trustworthy Secure Under Adversity." RONROSSECURE, LLC, August 2026. Available through the SSE Library at <https://ronrossecure.com>.
- [5] Ross, R. "Trustworthy Systems Need Trustworthy Parts." RONROSSECURE, LLC, July 2026. Available through the SSE Library at <https://ronrossecure.com>.