Trustworthy Systems Need Trustworthy Parts

Dr. Ron Ross
CEO, RONROSSECURE

Abstract

NIST SP 800-160, Volumes 1 and 2, describe a technology-agnostic discipline for engineering trustworthy, resilient systems built from composed elements. Several of the thirty security design principles in Volume 1, including Commensurate Trustworthiness, Substantiated Trustworthiness, Compositional Trustworthiness, and Minimal Trusted Elements, hold that system-level trustworthiness cannot be assumed; it must be established from evidence about those elements before any composition argument begins. Volume 2 reaffirms this in resiliency terms, naming component trustworthiness as a factor a resilient architecture must account for even while assuming eventual compromise. Neither volume specifies how a product's security claims are to be evidenced, but at several named points in its life cycle processes, Volume 1 points to the same external source: the Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408).

This paper argues a precise version of that relationship: SP 800-160 depends on substantiated component trustworthiness, not on the Common Criteria as its only possible evidentiary path, and the Common Criteria is the principal, standardized mechanism SP 800-160 itself cites for producing that substantiation. Drawing on the text in both publications, it traces this dependency through the trustworthiness context, Volume 2's resiliency techniques, and the specific points where SP 800-160 cites ISO/IEC 15408 by name. It maps six design principles to Common Criteria mechanisms, distinguishing direct correspondence from partial alignment; shows domain separation, the architectural core of defense against autonomous AI-driven attacks, is only as trustworthy as the components enforcing it; treats frontier AI as a component whose non-deterministic behavior resists that evaluation model; and addresses the standard's persistent cost and timeliness objection.

1. Introduction

NIST SP 800-160, Volume 1, Revision 1, *Engineering Trustworthy Secure Systems*, and NIST SP 800-160, Volume 2, Revision 1, *Developing Cyber-Resilient Systems*, together describe how to engineer a system, from stakeholder needs through operation and disposal, so that it is trustworthy and able to anticipate, withstand, recover from, and adapt to adversity. Both volumes are, by design, technology-agnostic. They describe a discipline for reasoning about trust, not a catalogue of specific products or a method for testing them. That is a considered choice, and it is the source of the discipline's durability. The same *problem* context, *solution* context, and *trustworthiness* context apply whether the system in question is built from mainframes or microservices.

That same technology-agnostic quality, however, creates a dependency the two volumes are explicit about even though neither volume resolves it directly. A *system*, in the NIST SP 800-160 context, is a composition of *system elements*. A system element can be a discrete component, product, service, subsystem, or human. The entire discipline rests on the ability to reason about the trustworthiness of those elements individually before reasoning about how they compose. Several of the thirty security design principles in Appendix E of SP 800-160 Volume 1, *Commensurate Trustworthiness*, *Minimal Trusted Elements*, *Substantiated Trustworthiness*, and *Compositional Trustworthiness* among them, state

this requirement plainly: **system-level trustworthiness cannot be assumed, it must be established from evidence about the constituent elements**. While SP 800-160 defines three processes (i.e., *Verification*, *Validation*, and *System Analysis*) that are used to generate assurance evidence during the system life cycle, it does not, by itself, provide a method for producing that evidence for a specific component product traditionally coming from the COTs marketplace. What it supplies is a citation. At each point in the engineering life cycle where a component's security requirements must be specified and its claims evaluated, Volume 1 cites the same source: ISO/IEC 15408, Parts 1 through 3, the *Common Criteria for Information Technology Security Evaluation*.

> ***"SP 800-160 does not, by itself, supply a method for producing evidence about a specific component product's trustworthiness. What it supplies is a citation."***

This paper argues a more precise claim than it might first appear to. The dependency is not that SP 800-160 requires the Common Criteria specifically, as though no other evidentiary path could satisfy it. The dependency is that SP 800-160 requires substantiated component trustworthiness, assurance evidence, not assumption, that a given system element actually possesses the security properties it is relied upon for, and the Common Criteria is the standardized mechanism SP 800-160 points to for producing that evidence. Where Common Criteria evaluation is unavailable for a given component category, the underlying requirement does not disappear; an organization must construct an equivalent evidentiary basis by other means. But where it is available, it is the mechanism SP 800-160 cites, repeatedly and at specific points in its own life-cycle processes, rather than one option among many left unstated. The two publications are not competitors and do not duplicate one another's work. SP 800-160 reasons about systems, and the Common Criteria specifies and evaluates the products those systems are built from. Removing rigorously evaluated evidence of component trust does not make SP 800-160 wrong; it removes the evidentiary floor the discipline needs to avoid what SP 800-160 itself calls *veneer security*: security functionality that appears to protect systems without the assurance evidence to back up that appearance.

The argument proceeds in eight further parts. Section 2 examines how Volume 1's own trustworthiness context and design principles state the composition problem. Section 3 shows that Volume 2 restates the same dependency in resiliency terms, naming component trustworthiness as one of the factors a resilient architecture must account for even while assuming eventual compromise. Section 4 describes what the Common Criteria actually is, how its structure answers the evidentiary need the first two sections identify, and where the bounds of a Common Criteria evaluation already sit. Section 5 traces the specific points in Volume 1's own life cycle process appendices where the Common Criteria is cited by name. Section 6 maps six of Volume 1's design principles to the Common Criteria mechanisms most relevant to each, distinguishing direct correspondence from looser forms of alignment, and then turns to domain separation, the architectural principle that contains an attacker once a component fails, to show that it depends on the same evaluated evidence this paper has been arguing for. Section 7 follows the dependency into acquisition and supply chain practice. Section 8 states plainly where the Common Criteria's evidentiary reach ends, including the sharper case of a frontier AI component whose behavior resists evaluation almost by construction. Section 9 addresses the standard's most persistent practical objection, cost and timeliness, and the degree to which AI-accelerated evaluation is beginning to change that calculus.

2. Systems Security Engineering's Composition Problem

The Trustworthiness Context and the Assurance Case

SP 800-160, Volume 1 organizes systems security engineering into three contexts. The *problem* context establishes what a system must achieve and what losses must be prevented. The *solution* context defines the protection strategy and architecture. The *trustworthiness* context is where the other two are demonstrated, through evidence, to actually hold. Volume 1 grounds the trustworthiness context in the concept of an *assurance case*: a structured set of arguments and a body of evidence showing that a system satisfies specific claims. An assurance case is only as strong as the evidence beneath it, and Volume 1 is candid that not all forms of evidence are equally strong. It describes three approaches to assurance, from weakest to strongest: (1) axiomatic assurance, belief accepted on faith or by demonstrated compliance; (2) analytic assurance, belief justified through testing and reasoning; and (3) synthetic assurance, belief built up from the structured composition of assurance about constituent parts.

Volume 1 explicitly warns against relying on the weakest of these. It names the failure mode veneer security: security functionality provided without corresponding assurance, so that the functionality only appears to protect resources when it does not. Compliance, the volume states directly, is a form of veneer security when used as the sole evidentiary basis for a trustworthiness judgment. This matters here because an assurance case built on components whose security properties have never been independently evaluated, only asserted by a vendor or assumed by an integrator, is exactly the axiomatic, faith-based assurance Volume 1 says is unsuited to complex systems. The trustworthiness context demands something better: credible, relevant evidence, produced by demonstration, inspection, analysis, and testing.

> ***"Assurance cases must be maintained... The specific form of an assurance case and the level of rigor and formality in acquiring the assurance evidence required is a trade space consideration."***

The Design Principles That Require Evidenced Component Trust

Several of Appendix E's thirty security design principles state the composition problem in different but converging terms. *Commensurate Trustworthiness* holds that a system element is trustworthy to a level commensurate with the most significant adverse effect that results from its failure. *Minimal Trusted Elements* holds that a system should have as few trusted elements as practicable, precisely because trusted elements are costlier to construct and require more analysis to qualify their trustworthiness. *Substantiated Trustworthiness* holds that trustworthiness judgments must be based on evidence, not assumption, and that the design mentality should be one of cautious mistrust, treating every element as untrustworthy until proven trustworthy. *Compositional Trustworthiness* holds, most directly of all, that the trustworthiness of an aggregate of composed system elements cannot be assumed from the trustworthiness assertions of each individual element, nor is it equal to the trustworthiness of the least trustworthy element. It is an *emergent property* that has to be separately determined.

Read together, these four principles describe a dependency chain rather than four independent rules. *Commensurate Trustworthiness* says the required level of component trust scales with consequence. *Minimal Trusted Elements* says to keep the set of trusted elements small so that establishing trust in each one is tractable. *Substantiated Trustworthiness* says that trust, once required, must be evidenced rather than assumed. *Compositional Trustworthiness* says that even fully evidenced, individually

trustworthy elements do not automatically yield a trustworthy system. The composition itself must be argued. None of the four principles specifies how a component's trustworthiness is actually to be evidenced, only that it must be. That is the gap the Common Criteria can help to close.

3. System Resiliency's Dependence on Component Trustworthiness

SP 800-160, Volume 2 extends the same dependency into the domain of system and cyber resiliency. Volume 2 organizes resiliency around four goals, *Anticipate*, *Withstand*, *Recover*, and *Adapt*, and around fourteen cyber resiliency techniques, several of which, including *Segmentation*, *Substantiated Integrity*, and *Analytic Monitoring*, presuppose that an organization can distinguish more trustworthy elements from less trustworthy ones. A segmentation strategy that separates critical functions from less critical ones is only as good as the organization's ability to correctly categorize which elements belong on which side of the boundary, a categorization that depends on evidence about each element's trustworthiness.

Volume 2 is also explicit that component trustworthiness, however well established, is not by itself sufficient. In its foundational assumptions for system and cyber resiliency, the publication states that a sophisticated adversary cannot always be kept out of a system or quickly detected and removed, despite the quality of the system design, the functional effectiveness of the security components, and the trustworthiness of the selected components. That statement is worth pausing on, because it does two things at once. It confirms that Volume 2 treats component trustworthiness as a real and necessary property, one of the three factors named alongside design quality and functional effectiveness. And it confirms that Volume 2 does not treat component trustworthiness as sufficient on its own; resiliency techniques exist precisely because even well-evidenced, well-selected components operate in complex systems where adversaries will always find some weakness, misconfiguration, or supply chain gap to exploit.

This is not a contradiction; it is the same layered logic in SP 800-160, Volume 1. Resiliency does not substitute for component trustworthiness, and component trustworthiness does not substitute for resiliency. Volume 2 reinforces the point through its treatment of supply chain risk, referencing SP 800-161 and recommending supply chain diversity as a resiliency technique precisely because organizations cannot fully verify the trustworthiness of every supplier and component in a complex, globally sourced system. Where a component's trustworthiness can be independently evaluated and evidenced, that evidence measurably narrows the residual uncertainty resiliency techniques otherwise have to absorb.

> ***"Resiliency does not substitute for component trustworthiness, and component trustworthiness does not substitute for resiliency."***

Resiliency Techniques That Presuppose a Trust Judgment

Three of SP 800-160 Volume 2's fourteen cyber resiliency techniques make the presupposition especially concrete. *Segmentation* defines and separates system elements based on criticality and trustworthiness, which requires a prior, defensible answer to the question of how trustworthy each element actually is before it can be assigned to a segment. *Substantiated Integrity* ascertains whether critical system components have been corrupted, on an ongoing basis, which requires a baseline understanding of what the uncorrupted, trustworthy state of that component looks like, an understanding that is far stronger when it rests on an evaluated system component than on an assumption about how the component is supposed to behave. *Analytic Monitoring* monitors and analyzes emergent system properties and behaviors on an ongoing, coordinated basis, which is only actionable if organizations

have a documented expectation of correct behavior to monitor against, the same documented expectation a system component's functional claims and the evaluation results are built to establish.

None of these three techniques requires Common Criteria evidence specifically, but each requires something functionally equivalent: a documented, credible basis for what a system component is supposed to do and how far it can be trusted, against which deviation or corruption can actually be detected. Where that basis is a vendor's unverified claim, the three techniques are only as reliable as the claim. Where it is Common Criteria evaluation evidence, produced by an accredited, independent laboratory against a published methodology, the same techniques inherit a materially stronger foundation without Volume 2 having to specify how that foundation is produced.

4. The Common Criteria: A Method for Specifying and Evaluating Component Trustworthiness

Before describing what the Common Criteria does, it is worth stating what it does not do, because the distinction shapes everything that follows. A Common Criteria evaluation evaluates a specifically defined *Target of Evaluation* against a specifically defined set of security claims and operating assumptions, recorded in a *Security Target*. It does not establish that a product is trustworthy in the abstract, in every environment, or for every purpose. Common Criteria Part 1 is direct about this stating that an evaluation result has meaning only in the context of the properties actually evaluated and the methods actually used, and an evaluated component deployed outside the assumptions of its own Security Target carries none of the assurance its certificate implies. This is not a weakness particular to the Common Criteria; it is a property of bounded evidence generally. But bounded evidence, honestly scoped and independently produced, is a fundamentally stronger foundation for a system-level trustworthiness argument than an unevaluated vendor claim, and it is the foundation SP 800-160 points to.

The Common Criteria, published as ISO/IEC 15408 and maintained through a multinational consortium that includes NIST and the National Security Agency, is built to do exactly what SP 800-160 assumes has already been done by the time a system element is composed into a system: specify the security properties an IT product claims to have, and independently evaluate whether it actually has them, within the bounds just described. The evaluation subject (Target of Evaluation or TOE) is a deliberately flexible entity that may be an entire IT product, a part of a product, a set of products, or a unique technology (e.g., a network device, an operating system, an application, an integrated circuit, or the cryptographic co-processor within one). The flexibility of the TOE concept is what allows evaluation at precisely the grain SP 800-160's *Minimal Trusted Elements* principle asks for: the trusted footprint an organization actually needs evaluated, no more and no less.

A *Protection Profile* expresses a reusable, implementation-independent set of security requirements for a category of product, such as firewalls or operating systems; a *Security Target* expresses the specific security requirements claimed for a particular product being evaluated. Both are built from two kinds of requirements defined in the Common Criteria. Security Functional Requirements, defined across eleven classes in Part 2, including *security audit*, *communication*, *cryptographic support*, *user data protection*, *identification and authentication*, *security management*, *privacy*, *protection of the TSF*, *resource utilization*, *TOE access*, and *trusted path or channels*, specify what security behavior the product must exhibit. Security Assurance Requirements, defined in Part 3 across classes covering *protection profile and security target evaluation*, *development evidence*, *guidance documentation*, *life-cycle support*, *testing*, *vulnerability assessment*, and *composition*, specify how much evidence must be produced and how rigorously it must be examined before a claim of conformance can be trusted. Assurance

components from these classes are packaged into predefined *assurance levels* in Common Criteria Part 5, giving acquirers, developers, and evaluators a common, internationally recognized scale for how much rigor a given claim of trustworthiness has actually earned.

The evaluation itself is performed by an accredited, independent third-party laboratory under a national scheme, using the *Common Methodology for Information Technology Security Evaluation* as its baseline procedure. This is precisely the analytic and synthetic assurance SP 800-160 Volume 1 describes as the stronger alternatives to axiomatic, faith-based assurance. Common Criteria Part 3 states its own purpose directly: it defines the assurance requirements from which the evaluation assurance levels and other packages are composed, and the criteria against which Protection Profiles, PP-Configurations, and Security Targets are themselves evaluated. A Common Criteria certificate is not a vendor's claim about its own product; it is an independently produced body of evidence, examined against a published, internationally recognized standard, that a specific product satisfies specific, published security claims. That is the evidentiary character Volume 1's trustworthiness context requires and that its assurance case concept is built to hold.

> ***"A Common Criteria certificate is not a vendor's claim about its own product. It is an independently produced body of evidence that a specific product satisfies specific, published security claims."***

Evidence Across the Product Life Cycle, Not Just at a Single Test

A recurring misunderstanding about the Common Criteria is that it certifies a product once, at a point in time, the way a certificate of compliance might. The assurance classes in Part 3 are built to avoid exactly that weakness. *Class ADV, Development*, requires evidence about the product's design at multiple levels of abstraction, from a functional specification of external interfaces down, at higher assurance levels, to implementation representation, so that an evaluator can trace security functional requirements to the mechanisms that actually enforce them rather than accepting a developer's description on faith. *Class ATE, Tests*, requires both developer testing evidence and independent evaluator testing, so that the claim of correct behavior is checked, not merely asserted. *Class AVA, Vulnerability Assessment*, requires an independent search for exploitable weaknesses and rates the attack potential needed to exploit any that are found, which is the closest the Common Criteria comes to directly testing the SP 800-160 design principle of *Self-Reliant Trustworthiness*: whether the component can protect itself independent of the correctness of everything around it. *Class ALC, Life-Cycle Support*, extends the evidence requirement backward into the developer's own configuration management, delivery, and flaw-remediation procedures, on the reasoning that a component built under an undisciplined development process cannot be fully trusted no matter how well its final release happens to test.

This life-cycle orientation is what makes Common Criteria evidence compatible with SP 800-160's own life-cycle framing rather than a one-time gate bolted onto it. SP 800-160, Volume 1 organizes its own life-cycle processes, including *Design Definition*, *System Analysis*, *Verification*, and *Validation*, around the idea that trustworthy secure systems engineering is sustained across the system's life, not established once at delivery and assumed to persist. Class ALC's requirements for developer flaw remediation and configuration management, and Class ACO's Composite design compliance family for TOEs that are later updated or recomposed, give an integrator continuing evidence to draw on precisely where SP 800-160's own system life-cycle processes ask for it.

Protection Profiles as Reusable Stakeholder Requirements

SP 800-160, Volume 1's early life-cycle process, *Stakeholder Needs and Requirements Definition*, exists to capture what stakeholders need protected and to transform that need into requirements a system solution must satisfy. Every organization that undertakes this process for a category of component it did not invent (e.g., an operating system, a firewall, a hardware security module), is at risk of re-deriving, in isolation and with uneven rigor, a set of security requirements that a community of similarly situated stakeholders has already worked out collectively. *Protection Profiles* exist to prevent exactly that duplication of effort. A Protection Profile is developed once, typically by a technical community, a user group, or a government agency, capturing the security functional and assurance requirements that are appropriate to an entire category of component products, and it is then reused across every acquisition and every vendor's *Security Target* for that category. This gives Volume 1's *Stakeholder Needs and Requirements Definition* process access to a body of pre-vetted, community-tested requirements for common component categories, rather than requiring each acquiring organization to reconstruct that expertise independently, which is precisely the kind of reusable engineering discipline Volume 1's own reference to ISO/IEC 15408 at the stakeholder-requirements stage anticipates.

5. Where SP 800-160 Points Directly to the Common Criteria

The dependency argued for in this paper is not an inference drawn from general compatibility between the two publications. SP 800-160, Volume 1 cites ISO/IEC 15408, Parts 1 through 3, by reference at specific points in its system life cycle processes, and those points are not incidental. They are the specific tasks in which a security requirement is transformed from a stakeholder need into a specification, a design is evaluated against that specification, system analysis results are produced and recorded, and a component product or service is formally evaluated for acceptance. Table 1 lists the system life cycle processes in Volume 1 that cite the Common Criteria directly.

Table 1: SP 800-160, Volume 1 Life Cycle Processes Citing ISO/IEC 15408

SP 800-160 Life Cycle Process	Task
Stakeholder Needs and Requirements Definition	**SN-4:** Identifies the security-relevant constraints on a system solution and defines stakeholder requirements in a way that's consistent with those security aspects and protection needs — the step where a protection need first becomes a stated requirement.
System Requirements Definition	**SR-2:** Defines each security function the system must perform (across all states, modes, and transitions), the security aspects of every other function, necessary security-driven implementation constraints, and the system security requirements themselves, with supporting rationale and traceability metadata.
Design Definition	**DE-2:** Allocates security requirements to system elements, transforms security-relevant architectural entities into design elements and trustworthy secure design characteristics, refines the security aspects of interfaces between elements, and develops the security aspects of design artifacts (specifications, data sheets, configuration and procedure documentation).
	DE-4: Obtains agreement on the security aspects of the design, maps the trustworthy secure design characteristics to specific system elements, records design decisions and rationale, and maintains traceability from those characteristics back to requirements, interfaces, analysis results, and verification methods.

SP 800-160 Life Cycle Process	Task
System Analysis	**SA-2:** Applies selected analysis methods to the security-relevant aspects of the system, reviews the results for quality and validity (coordinating with prior analyses), establishes conclusions and recommendations, and records the results.
	SA-3: Maintains bidirectional traceability among the analysis results, the methods used, the data and assumptions relied on, and the problem the analysis was addressing, and provides the security-relevant artifacts selected for project baselines.
Quality Assurance	**QA-1:** Defines the security aspects of the quality assurance strategy — roles, responsibilities, life-cycle-specific and supplier-specific security activities, required verification/validation/test activities, and acceptance criteria — and establishes the independence of security quality assurance from the other life-cycle processes.

The pattern across these five citation points is consistent. Every one of them is a task where SP 800-160 requires an organization to produce or evaluate evidence about a security requirement or a system element, and at every one of them, Volume 1 points outward to the Common Criteria rather than specifying its own evaluation method. This is consistent with Volume 1's own stated scope: it is a systems security engineering framework, not a product evaluation methodology, and the publication clearly did not intend to duplicate a mature, internationally recognized evaluation standard that already existed for that purpose. The citation structure effectively directs the systems engineer to an established external evaluation methodology for addressing the component-evidence problem.

6. Mapping the Design Principles to Common Criteria Mechanisms

Section 2 identified four design principles, *Commensurate Trustworthiness*, *Minimal Trusted Elements*, *Substantiated Trustworthiness*, and *Compositional Trustworthiness*, that describe requirements for evidenced component trust without specifying how that evidence is to be produced. Two further principles, *Least Functionality* and *Self-Reliant Trustworthiness*, make the same demand in narrower form.[1] Table 2 maps each of these six principles to the Common Criteria mechanism most relevant to it, but the fit is not uniform across all six, and claiming otherwise would overstate the case. Some rows reflect a direct correspondence, where the Common Criteria mechanism addresses precisely the scoping or evidentiary decision the principle describes. Others reflect a strong alignment, where the mechanism produces the right kind of evidence but the principle's full judgment still has to be made by the organization applying it. A smaller number reflect only a partial or conditional alignment, where the Common Criteria artifact is a necessary input but not, by itself, sufficient to satisfy the principle. The table's fourth column marks each row accordingly.

[1] The security design principles listed in this section are not exhaustive and are used only as illustrative examples.

Table 2: Mapping SP 800-160 Security Design Principles to ISO/IEC 15408 Mechanisms

Design Principle	What the Principle Requires	ISO/IEC 15408 Mechanism	Alignment
Commensurate Trustworthiness	A system element must be trustworthy to a level commensurate with the worst consequence of its failure.	Assurance packages (composed into EALs) let an acquirer select a level of rigor scaled to the consequence of component failure.	**Partial/Conditional** EALs scale the rigor of evaluation, but selecting one is not the same decision process as SP 800-160's consequence analysis.
Minimal Trusted Elements	A system should rely on as few trusted elements as practicable, since trusted elements cost more to build and analyze.	The Target of Evaluation boundary in a Security Target scopes exactly which functions are being trusted and evaluated, keeping the trusted footprint explicit and minimal.	**Direct** TOE boundary is the same scoping decision the principle asks for.
Substantiated Trustworthiness	Trustworthiness judgments must rest on evidence, not assumption; the design mentality is cautious mistrust until proven otherwise.	Security Assurance Requirements including development evidence (ADV), testing (ATE), and independent vulnerability assessment (AVA), are the evidence base an assurance case needs.	**Strong** SARs are evidence-producing by design, though the assurance case itself must still be built from that evidence.
Compositional Trustworthiness	The trustworthiness of a system cannot be assumed from the trustworthiness assertions of its individual elements; it must be established for the aggregate.	Class ACO (Composition) defines assurance requirements specifically for establishing confidence in a TOE built from previously evaluated components.	**Direct** ACO exists specifically to address the case this principle describes.
Least Functionality	A system element should do only what it needs to do, and COTS components — which typically contain excess functionality — must be configured down to that scope.	Security Functional Requirements selected in a Protection Profile or Security Target define exactly which functions are in scope.	**Partial/Conditional** ST defines evaluated functionality; it does not itself confirm unneeded functions were disabled in deployment.
Self-Reliant Trustworthiness	A trustworthy element must be able to protect itself independent of the correct behavior of other elements.	AVA_VAN (vulnerability analysis) and the associated attack-potential rating test whether a component resists compromise on its own, independent of surrounding controls.	**Strong** AVA_VAN tests self-reliance directly, within the attack-potential level the evaluation targeted.

The direct-correspondence rows follow closely from how the Common Criteria was constructed. The TOE-boundary concept exists so that an evaluation, and the trust an acquirer places in its result, is scoped to a specific, bounded claim rather than an entire product's unexamined functionality, which is the same discipline that SP 800-160, Volume 1's *Minimal Trusted Elements* asks of a system architect deciding which elements deserve trusted status. Composition assurance, *Class ACO*, exists specifically to address the case *Compositional Trustworthiness* describes: a TOE built from other, already-evaluated

TOEs, where the trust in the whole cannot simply be inherited from the trust in the parts and must instead be separately argued and evidenced.

The partial-alignment rows deserve equal attention, because overstating them is exactly the kind of claim that weakens an otherwise defensible argument. *Commensurate Trustworthiness* asks that a component's required trust level scale with the consequence of its failure to the mission; an *Evaluation Assurance Level* scales the rigor and depth of the evaluation itself, which is a related but distinct decision, made by an evaluation scheme and a *Protection Profile* author rather than derived from a specific system's consequence analysis. An organization still has to perform its own commensurate trustworthiness judgment and then select an EAL, or an equivalent evidentiary target, that satisfies it. The EAL does not perform that judgment on the organization's behalf. *Least Functionality* is similar: a Security Target's Security Functional Requirements document precisely which functions were evaluated, which gives an integrator the information needed to disable the rest, but the ST does not verify that a specific deployment actually disabled them. The evidentiary input is real; the last step of applying it remains the acquiring organization's responsibility.

Least Functionality deserves a further note because Volume 1 raises the Common Criteria's central subject matter, commercial off-the-shelf components, directly in its own discussion of the principle. Volume 1 observes that COTS components typically contain functions beyond those required to fulfill their intended purpose, and that where it is not possible to avoid this, the component should be configured to enable only the required functions and to prohibit or restrict the rest. That guidance assumes an engineer can determine, for a specific COTS product, which functions are actually required and which are surplus, a determination the product's own documentation rarely makes easy or verifiable on its own. A Security Target's explicit enumeration of Security Functional Requirements, scoped to a defined TOE boundary, gives an integrator a documented, evaluated basis for making that determination. It is an input to Volume 1's caution about COTS components, not a substitute for the configuration step itself. An integrator still has to act on the Security Target's enumeration by actually disabling what falls outside it, and the evaluation evidence cannot confirm on its own that a specific deployment did so.

> ***"SP 800-160, Volume 1's design principles describe what a trustworthy composition requires of its parts. For many of those requirements, the Common Criteria is where the claim becomes testable and where it is actually tested, although the organization applying the principle still has to close the last step itself."***

Domain Separation: Where Architectural Containment Meets Component Assurance

The six principles mapped above describe how a single component earns trust. A separate line of argument, developed in a companion analysis of AI-accelerated cyberattacks, addresses how an architecture built from many such components contains the damage when one of them fails anyway. That analysis identifies the design principle of *Domain Separation* as the bedrock structural defense against autonomous, machine-speed adversaries. A properly domain-separated architecture confines an initial compromise to the domain in which it occurred, forcing an attacker to convert an inexpensive foothold into mission impact the hard way, one bounded domain at a time, rather than moving freely through a flat, undifferentiated network. *Least Privilege*, *Mediated Access*, *Least Functionality*, *Least Sharing*, *Least Persistence*, *Structured Decomposition and Composition*, *Continuous Protection*, and *Anomaly Detection* are identified there as the principles that make domain separation implementable and durable rather than aspirational.

That argument and this paper's argument meet at a point worth making explicit. Domain separation is not self-enforcing; it is enforced by specific system elements (e.g., firewalls, cross-domain solutions, data diodes, software-defined network controllers, hardware-enforced isolation mechanisms, and mandatory access control implementations), and each of those elements is itself a component whose trustworthiness has to be substantiated before the domain boundary it enforces can be trusted. An organization can adopt every architectural recommendation in that analysis and still be relying on axiomatic assurance if the boundary-enforcement components themselves carry no independent evidence of their security properties. This is precisely the gap the Common Criteria exists to close, and, unlike the coverage gaps noted elsewhere in this paper, it is a gap the Common Criteria already covers reasonably well: network devices, firewalls, and cross-domain solutions are among the most mature Common Criteria evaluation categories, with various Protection Profiles built specifically around the containment and mediated-access properties that domain separation requires.

> ***"Domain separation is not self-enforcing. It is enforced by specific components — and each of those components is itself a system element whose trustworthiness has to be substantiated before the boundary it enforces can be trusted at all."***

The security design principle, *Mediated Access*, provides the clearest illustration. The reference monitor concept described in that analysis, a policy-based decision point that every domain-crossing interaction must pass through, tamper-resistant, always invoked, and small enough to verify, is not a new idea introduced for AI-driven threats; it is a design target the Common Criteria's own assurance classes were built to test. A Security Target's functional requirements specify what the reference monitor is supposed to enforce; the vulnerability assessment activities in Class AVA test whether the ST actually resists compromise; and the development evidence in Class ADV lets an evaluator trace the claimed access-mediation behavior down to the mechanism that implements it. Domain separation, in other words, is an architectural conclusion whose premises are exactly the kind of component-level claims this paper has argued need Common Criteria evidence, or its equivalent, to be more than assertion.

7. Acquisition and Supply Chain: Where SP 800-160 and CC Meet in Practice

The dependency this paper describes is not only conceptual. It surfaces directly in how SP 800-160 addresses acquisition. Volume 1's *Design Definition* process instructs an engineer to determine the security technologies required for each system element composing the system and to identify the security concerns associated with each technology, explicitly including concerns arising from known and potential vulnerability within or enabled by the supply chains involved in acquiring those technologies. That instruction presumes an organization has some way to specify, in a solicitation or contract, what security properties an acquired component must have, and some way to verify, before or after delivery, that the delivered component actually has them. A *Protection Profile* or *Security Target* is precisely that specification mechanism, expressed in a vendor-neutral, internationally recognized form that can be written into a requirements document or a statement of work rather than invented from scratch by each acquiring organization.

SP 800-160, Volume 2 reinforces the same point from the resiliency side. Its treatment of supply chain risk management, grounded in SP 800-161, and its recommendation of supply chain diversity both assume an organization can meaningfully distinguish among suppliers and components on the basis of trustworthiness, not merely price or availability. An organization that cannot articulate what trustworthy means for a component category has no basis for that distinction beyond reputation and hope. Common Criteria evaluation gives an acquisition process a documented, third-party-verified basis for exactly that

distinction, and gives supply chain risk analysis a concrete artifact, the *Security Target* and its evidence, to assess rather than a marketing claim.

> ***"An organization that cannot articulate what trustworthy means for a component category has no basis for that distinction beyond reputation and hope."***

None of this implies that Common Criteria evaluation is available or practical for every component. Evaluated products exist in some categories and not others, and evaluation takes time and cost not every acquisition can absorb. Where an equivalent evaluated product does not exist, the discipline the Common Criteria embodies, an explicit, testable specification of required security properties and independent evidence that a candidate product meets it, remains the right acquisition posture even without a certificate to show for it.

8. The Limits of the Dependency

Section 4 already flagged the central limit: a Common Criteria evaluation certifies a bounded claim about a bounded TOE, not universal trustworthiness. The rest of Common Criteria Part 1's own scope statement reinforces the same caution from other directions. The Common Criteria does not contain evaluation criteria for administrative security measures that are not directly related to IT security functionality, even though it recognizes that organizational, personnel, physical, and procedural controls contribute significantly to real-world security. It does not address the legal or administrative framework under which an evaluation scheme operates, nor the accreditation process by which authority is granted to operate a product in its full operational environment, including its non-IT parts. It does not itself assess the inherent mathematical properties of cryptographic algorithms, leaving that to the evaluation scheme under which the Common Criteria is applied.

These limits are precisely why SP 800-160 remains necessary and is not itself redundant with the Common Criteria. A system composed entirely of Common Criteria-evaluated components is not automatically a trustworthy system in the SP 800-160 sense; *Compositional Trustworthiness* still has to be separately argued, the operational environment and its non-IT parts still have to be accounted for, and Volume 2's cyber resiliency techniques still have to compensate for the adversary presence that even well-evaluated components cannot categorically exclude. The Common Criteria supplies rigorous, independently produced evidence about the parts. SP 800-160 supplies the discipline for reasoning about the whole, including everything the Common Criteria explicitly declines to cover. Each publication is necessary; neither is sufficient alone, and the dependency runs in only one direction that matters for this paper's argument. SP 800-160's assurance case cannot be built on unevaluated components without collapsing into the axiomatic, veneer-security judgment Volume 1 itself warns against, while the Common Criteria's product-level evaluations do not require SP 800-160's system-level reasoning to remain valid.

A further limit worth naming is coverage. Common Criteria evaluation is mature and widely available for some product categories, network devices, operating systems, mobile device management, hardware security modules, and comparatively thin for others, including many of the AI-enabled components an SP 800-160-grounded architecture increasingly has to compose. Where an evaluated product does not exist, an organization is not excused from the underlying requirement; *Substantiated Trustworthiness* still demands evidence, not assumption, and the acquiring organization has to construct an equivalent evidentiary basis, through tailored testing, independent assessment, or a negotiated set of supplier-provided evidence, rather than falling back to the vendor's own unverified assertion. The absence of a

Common Criteria certificate for a given component category is a gap in available assurance mechanisms, not a license to treat *Commensurate Trustworthiness* and *Substantiated Trustworthiness* as satisfied by default.

The Sharpest Case: A Component That Resists Evaluation

Frontier AI models embedded as system components are worth treating as a case on their own, because they do not merely add to the coverage gap just described; they test whether the Common Criteria's evaluation model applies to this category of component at all. A companion analysis of frontier AI's role in mission-resilient systems makes the underlying reason precise rather than casual. Any physical computer, including one running a frontier AI model, has finite memory and is therefore, in the strict formal sense, a finite state machine, and a finite state machine with a fixed transition rule for every state and input is deterministic by definition. But almost nothing about a deployed AI system holds those conditions fixed in practice: sampling temperature and random seeds introduce deliberate randomness, floating-point operations on parallel hardware can execute in different orders from run to run, concurrent execution introduces race conditions, and real deployments take in external inputs that are not identical from one invocation to the next. The practical result is that identical prompts can produce different outputs across runs of the same model.

Common Criteria evaluation, at every assurance level, is built around the assumption that a TOE's behavior can be specified precisely enough to test against: a *Security Target* states functional claims, and *Class ADV* and *Class ATE* evidence demonstrates that the implementation satisfies those claims consistently. The difficulty is that the security-relevant behavior of frontier AI is difficult to specify exhaustively, bound, and evaluate against a stable Security Target — not only that its outputs vary. This is the same difficulty that Section 2 traced through *Minimal Trusted Elements* and *Compositional Trustworthiness* applied to conventional software; frontier AI components make it acute rather than introducing it. An organization that treats a frontier model as a classically evaluated, trusted component because it passed a benchmark or a red-team exercise once is making a claim the component's own behavior does not support, for exactly the reason a Common Criteria certificate would not support it either.

> ***"The difficulty is that the security-relevant behavior of frontier AI is difficult to specify exhaustively, bound, and evaluate against a stable Security Target."***

The correct posture, consistent with both this paper's argument and the companion analysis, is not to wait for an evaluation model that may never arrive in its current form. It is to treat a frontier AI component the way SP 800-160, Volume 2's *Assume Compromised Resources* principle already treats any component that cannot be trusted to behave as intended: bound its authority, access, and potential impact architecturally, so that the system's trustworthiness case does not depend on the component's behavior being fully specified. *Domain Separation*, discussed in Section 6, is one of the more direct ways to do this: isolating a frontier AI component behind a mediated, evaluated boundary substitutes an evidenced claim about the boundary for an unevidenced claim about the model, which is the same substitution this paper has argued for throughout, applied to the one component category where direct evaluation is least tractable today.

9. Answering the Practical Objection: Cost, Timeliness, and What AI Is Changing

Everything argued so far in this paper takes for granted that Common Criteria evaluation is a practical mechanism an organization can actually use. That assumption deserves to be tested directly, because the standard's most persistent real-world criticism is not conceptual, it is economic. A rigorous evaluation at the upper *Evaluation Assurance Levels* can take eighteen to twenty-four months and cost hundreds of thousands to millions of dollars. By the time a certificate issues, the product itself may be two or three releases past the version that was actually evaluated. The pool of evaluators qualified to do this work is small, specialized, and expensive, and in a software ecosystem that ships updates in hours, a certification model built around multi-year cycles struggles to keep pace with the very threat landscape it exists to address. As I have argued elsewhere, the Common Criteria approach is not wrong; the impediments have been practical.

The urgency here is not abstract. A companion analysis of next-generation AI-driven cyberattacks describes what has come to be called a Mythos-class attack: autonomous zero-day exploitation, multi-stage autonomous attack chains, and rapid, low-cost exploitation, carried out by AI models capable of independently scanning code, locating vulnerabilities, and generating functional exploit code with minimal human direction. Whatever label an organization uses for this category of threat, the economics are the relevant point for this section's argument: the same acceleration that compresses an adversary's reconnaissance-to-exploit timeline from weeks to hours has no counterpart on the assurance side unless evaluation itself accelerates too. An evaluation model built around eighteen-to-twenty-four-month cycles was never going to keep pace with a human-speed adversary; it has even less chance of keeping pace with one operating at machine speed. That mismatch is precisely what the rest of this section addresses.

It is worth being precise about where the cost actually accumulates, because the answer matters for whether the problem is fixable. A Common Criteria evaluation is structured around three ascending dimensions of effort (i.e., scope, depth, and rigor), and all three expand together at higher EALs. Historically, every dimension has required significant human labor: an evaluator reads and analyzes extensive *Security Target* documentation, traces security functional requirements through design documents to implementation, develops and executes test cases against the *Target of Evaluation*, examines development-environment controls, and produces formal evaluation reports for every finding. Three structural problems compound that labor cost: (1) a documentation burden that consumes vendor resources to create and, worse, to maintain across product versions; (2) a genuine scarcity of evaluators who combine security engineering depth, Common Criteria framework knowledge, and domain-specific product expertise; and (3) a point-in-time rigidity in which any change to a product under evaluation triggers re-evaluation, creating a disincentive to update the product at all and letting evaluated configurations drift away from what is actually deployed. The result is a framework with high integrity and low accessibility: organizations with regulatory mandates and deep resources pursue evaluation, and most of the rest of the ecosystem does not.

Artificial intelligence does not solve the underlying requirement that assurance be grounded in evidence rather than assertion, and it does not shrink the EAL structure itself. What it can do is compress the labor cost inside each of the three dimensions that drive that structure. Modern AI systems, large language models paired with static analysis, fuzzing engines, and formal verification tools, can examine implementation artifacts at a depth and speed no human evaluation team can match, addressing directly the verification burden, confirming that a product does what its documentation claims and nothing more, that consumes so much evaluator effort at the higher EALs. The same tools can accelerate

Security Target preparation by drafting conformant content from existing product documentation and prior STs for similar product types, and, more significantly, can continuously check that the security objectives fully address the security problem definition and that functional requirements trace correctly, turning a periodic manual milestone into an ongoing automated check. AI-enabled offensive security tools can systematically probe attack surfaces and chain exploitation attempts at a scale and consistency manual penetration testing cannot reach, which does not just lower the cost of the vulnerability assessment activities that anchor the highest EALs, it broadens their coverage, and broader coverage is more credible assurance, not merely less expensive assurance.

The structurally most significant change is the shift from point-in-time evaluation to continuous evaluation. If the conformance checking, vulnerability analysis, documentation review, and test execution behind a Common Criteria evaluation can be automated and run continuously, then a product update no longer has to trigger a multi-month re-evaluation project; it can trigger an automated analysis of the delta, what changed, how the change affects the security architecture, whether functional requirements still hold, whether new vulnerabilities were introduced, with human evaluators reviewing that delta assessment and making the judgments that remain genuinely theirs to make. That alignment of evaluation cadence with development cadence is precisely the mismatch this paper's Section 8 named as a coverage gap: evaluated products are comparatively thin for the fast-moving component categories, including AI-enabled ones, that a modern SP 800-160-grounded architecture increasingly has to compose. An evaluation model that can turn around in days rather than years does not eliminate that gap by itself, but it removes the economic reason the gap exists in most product categories today.

> ***"We need more trustworthy secure products so that we can build more trustworthy secure systems. That is not a statement about compliance. It is about a foundational dependency between component-level assurance and system-level trustworthiness that SP 800-160 makes explicit and CC evaluation is uniquely positioned to satisfy."***

None of this changes the paper's central claim; if anything, it sharpens it. AI accelerates the production of evidence. It does not, and by design cannot, replace the structured reasoning that turns evidence into an assurance case, the human judgment about whether evidence is relevant, credible, and sufficient to support a trustworthiness claim, which is exactly the interpretive step SP 800-160's *trustworthiness context* requires and no automated tool performs on its own. There is also a recursive version of this paper's own argument that the field has to confront directly: if AI tools are themselves conducting evaluation activities, the trustworthiness of those AI tools becomes a relevant question, and an AI system that generates flawed test cases or fails to surface a real vulnerability can produce evidence that looks comprehensive while being systematically incomplete. The same evidentiary discipline this paper has argued the Common Criteria brings to IT products in general has to be applied to the AI systems increasingly used to produce that evidence. That is not a reason to set AI-accelerated evaluation aside; it is a reason to treat it as one more system element whose own trustworthiness has to be substantiated before its output is trusted, which is simply this paper's thesis applied one level down.

For the argument this paper has made, the practical upshot is direct. The most common objection to relying on the Common Criteria as SP 800-160's principal cited evidentiary mechanism, that the standard is too slow and too expensive to matter for most of the components a real system is actually built from, is a real objection about today's economics, not a permanent limit on the mechanism itself. If AI-accelerated evaluation closes even a fraction of the cost and timeline gap I have described elsewhere, the library of components carrying independently evidenced, rather than merely asserted, security

properties expands accordingly, and *Compositional Trustworthiness* arguments at the system level get to rest on a correspondingly larger, and correspondingly more current, evidentiary base.

10. Conclusion

NIST SP 800-160 is, by its own account, a discipline for reasoning about trust in systems built from composed elements. That discipline succeeds only where the elements being composed are themselves demonstrably trustworthy, a requirement Volume 1 states directly through the security design principles of *Commensurate Trustworthiness*, *Minimal Trusted Elements*, *Substantiated Trustworthiness*, and *Compositional Trustworthiness*, and that Volume 2 reaffirms by naming component trustworthiness, alongside design quality and functional effectiveness, as one of the factors a resilient architecture has to account for even while assuming an adversary will eventually get in. The precise claim this paper has argued is not that SP 800-160 requires the Common Criteria as its only possible evidentiary path. It is that SP 800-160 requires substantiated component trustworthiness, full stop, and that the Common Criteria is the standardized mechanism SP 800-160 points to, at specific named points in its own life-cycle processes, for producing that substantiation. That is a narrower claim than "SP 800-160 depends on the Common Criteria," and a more defensible one: it survives the fact that evaluated products do not exist for every component category, and it survives the fact that a Common Criteria certificate, honestly read, only ever certifies a bounded claim about a bounded TOE.

The practical implication for an organization applying the *Multidimensional Protection Strategy*, or any systems security engineering effort grounded in SP 800-160, is that the architecture's design principles and its evidentiary floor are two different deliverables, and both are required. Selecting Common Criteria-evaluated products where they exist, and where they do not, specifying Protection Profile-equivalent requirements and demanding comparable independent evidence from suppliers, is not a separate compliance exercise running alongside the systems engineering effort. It is how the assurance case Volume 1 requires acquires the credible, relevant evidence it cannot manufacture on its own. A system engineered to SP 800-160's principles but composed of components whose trustworthiness was never independently evaluated, by the Common Criteria or an equivalent, is not a trustworthy system with a paperwork gap. It is, in Volume 1's own words, veneer security: an architecture that looks like it satisfies *Compositional Trustworthiness* because no one has yet produced the evidence that it does not.

> ***"A system composed of components whose trustworthiness was never independently evaluated is not a trustworthy system with a paperwork gap. It is veneer security."***

Neither SP 800-160 nor the Common Criteria replaces the engineering judgment of the people applying them, and the mapping in Section 6 was built deliberately to say so: some Common Criteria mechanisms correspond directly to a design principle's requirement, others only partially close the gap, leaving a final judgment the organization still has to make. A Common Criteria certificate does not certify the system an evaluated component is ultimately deployed into, and SP 800-160's design principles do not, by themselves, tell an engineer which product to buy or which residual step still needs verifying by hand. What these publications do together is close a loop that neither closes alone: SP 800-160 requires that component trustworthiness be evidenced before it can be composed and reasoned about at the system level, and the Common Criteria is the standard most explicitly built, and most explicitly cited by SP 800-160 itself, to produce a substantial share of that evidence, at the grain, the rigor, and the independence the requirements call for.

This argument does not stand apart from the broader case for structural security against AI-accelerated threats; it is the evidentiary layer underneath it. The security design principle of *Domain Separation*

contains an attacker once a component fails, but only if the components enforcing that separation are themselves substantiated rather than assumed. Frontier AI embedded as a system component tests the limits of what evaluation can currently certify, but the answer is not to abandon the requirement for evidence; it is to architect around the component's unevaluated behavior the same way this paper has argued components should be treated when no evaluation exists. Component-level assurance, architectural containment, and AI-specific governance are three layers of the same discipline, and none of the three substitutes for the other two.

References

Ross, R., Winstead, M., & McEvilley, M. (2022). NIST Special Publication 800-160, Volume 1, Revision 1: Engineering Trustworthy Secure Systems. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-160v1r1

Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., & McQuaid, R. (2021). NIST Special Publication 800-160, Volume 2, Revision 1: Developing Cyber-Resilient Systems: A Systems Security Engineering Approach. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-160v2r1

Common Criteria Recognition Arrangement Members. (2022). Common Criteria for Information Technology Security Evaluation, CC:2022, Revision 1 — Part 1: Introduction and General Model. https://www.commoncriteriaportal.org/files/ccfiles/CC2022PART1R1.pdf

Common Criteria Recognition Arrangement Members. (2022). Common Criteria for Information Technology Security Evaluation, CC:2022, Revision 1 — Part 2: Security Functional Components. https://www.commoncriteriaportal.org/files/ccfiles/CC2022PART2R1.pdf

Common Criteria Recognition Arrangement Members. (2022). Common Criteria for Information Technology Security Evaluation, CC:2022, Revision 1 — Part 3: Security Assurance Components. https://www.commoncriteriaportal.org/files/ccfiles/CC2022PART3R1.pdf

Ross, R. (2026). AI and the Common Criteria: Making Assurance Timely, Accessible, and Continuous. https://www.linkedin.com/posts/ronrossecure_ai-and-the-common-criteria-activity-7479670333701074944-EcHf

Ross, R. (2026). Defending Against Mythos-Class Attacks: Applying Structural Design Principles to Build Trustworthy Secure Systems. https://www.linkedin.com/pulse/defending-against-mythos-class-attacks-ron-ross-pgqxe/?trackingId=P8f49%2BCxRz6jOuSRFvpSuw%3D%3D

Ross, R. (2026). A Multidimensional Protection Strategy for Building Mission-Resilient Systems in the Age of Frontier AI. https://www.linkedin.com/pulse/multidimensional-protection-strategy-building-systems-ron-ross-q8a3e/?trackingId=P8f49%2BCxRz6jOuSRFvpSuw%3D%3D

Boyens, J. M., Paulsen, C., Moorthy, R., & Bartol, N. (2015). NIST Special Publication 800-161: Supply Chain Risk Management Practices for Federal Information Systems and Organizations. National Institute of Standards and Technology. https://csrc.nist.gov/pubs/sp/800/161/r1/upd1/final

