ASSURANCE AND TRUSTWORTHY SECURE SYSTEMS

Why Evidence Is the Basis for Trust

Dr. Ron Ross

CEO, RONROSSECURE, LLC
ISTS Fellow, Dartmouth College

Abstract

Trustworthy secure systems cannot be established by assertion, reputation, compliance, or the presence of security functionality alone. NIST Special Publication 800-160, Volume 1, Revision 1 makes assurance the evidentiary foundation for trustworthiness: assurance provides the grounds for justified confidence that security claims have been or will be achieved, and that confidence is derived from objective evidence that reduces uncertainty to an acceptable level. This paper develops assurance as the unifying concept across trustworthy secure design, adequate security, engineering for compromise, multidimensional protection, component trustworthiness, resilience, and the broader systems security engineering discipline. It distinguishes the concept of trust from trustworthiness; explains why trustworthiness exists on a continuum and why the required level of trust must be commensurate with consequence; examines axiomatic, analytic, and synthetic approaches to assurance; and shows how an assurance case connects security objectives, claims, arguments, evidence, assumptions, and residual uncertainty. Particular attention is given to the challenge of opaque or black-box components, where limited visibility constrains the evidence that can be obtained and therefore constrains the trustworthiness that can responsibly be claimed. The central proposition is simple: trust should never exceed the assurance that the available evidence can justify.

1. Introduction

The language of cybersecurity is saturated with the term *trust*. We trust identities, devices, cloud services, software, administrators, suppliers, artificial intelligence models, cryptographic modules, recovery systems, and the architectures that compose them. Yet trust itself is not a security property. Trust is a *decision* to depend on an entity. The engineering question is whether that dependence is *justified*.

NIST SP 800-160, Volume 1, Revision 1 provides a disciplined answer. It distinguishes the concept of trust from *trustworthiness* and grounds trustworthiness in the concept of *assurance* [1]. A trustworthy entity requires sufficient evidence to support the claims for which it is relied upon [2][3]. Assurance, in turn, is the grounds for justified confidence that a claim or set of claims has been or will be achieved [5]. The relationship is therefore not semantic. It is *causal*: evidence supports assurance; assurance supports justified confidence; justified confidence supports a trustworthiness judgment; and that judgment informs how much trust should be granted.

> ***"By trustworthiness, we mean simply worthy of being trusted to fulfill whatever critical requirements may be needed for a particular component, subsystem, system, network, application, mission, enterprise, or other entity."*** **[4]**

This relationship is the connective tissue across systems security engineering. A security design principle is useful only if there is evidence that its intended property has actually been realized. For example, a compromise containment boundary is meaningful only if there is justified confidence that the boundary cannot be bypassed in the states that matter. An adequately secure system is not one that merely appears to have enough security controls; it is one for which the adequacy judgment can be defended. A resilient

recovery mechanism is not trustworthy merely because it restores service; there must be evidence that it restores an authorized and sufficiently trustworthy secure state. Component trustworthiness cannot be inherited from a vendor claim, and system trustworthiness cannot be inferred merely by assembling components that are individually described as trustworthy secure.

Trust should never exceed the assurance that available evidence can justify.

Assurance is therefore not an appendix to systems security engineering. It is what separates a security claim from a security belief. It is the mechanism by which an organization answers the question that follows every important architectural assertion: How do you know?

2. Trust, Trustworthiness, and Assurance Are Different Concepts

SP 800-160 treats trust and trustworthiness as related but distinct. Trust is a belief that an entity will meet certain expectations and can therefore be relied upon. Trustworthiness is a judgment that the entity is worthy of that trust, based on evidence supporting the relevant claims [1]. An organization can trust an entity without knowing whether it is trustworthy.[1] That happens constantly in complex systems because there may be no practical alternative, the dependency is hidden, or the evidence needed to evaluate the dependency is unavailable.

That distinction matters because architecture creates trust dependencies whether engineers acknowledge them or not. If failure or malicious behavior of a component can invalidate a security claim, that component is trusted for that claim. The system may depend on an identity provider to authenticate administrators, a hypervisor to isolate workloads, a cryptographic module to protect keys, a sensor to report physical state, or a recovery service to restore software. Calling those components “trusted” says nothing about whether they deserve the trust placed in them. It only identifies the dependency.

Trustworthiness asks the harder question: what evidence supports the expectation? Assurance answers how confidence in that evidence and the resulting claim is established. SP 800-160 states that justified confidence is derived from objective evidence that reduces uncertainty to an acceptable level. The evidence must be relevant, accurate, credible, and sufficient in quantity to support reasoned conclusions. It is produced through engineering methods that include demonstration, inspection, analysis, evaluation, testing, verification, and validation [1]. The resulting relationship can be expressed as an engineering *chain*:

Security objective → claim → evidence → argument → justified confidence → trustworthiness judgment → decision to trust.

Breaking any link weakens the conclusion. A claim without evidence is an assertion. Evidence without a clear claim may be technically interesting but decisionally irrelevant. Evidence and claims without a valid argument do not establish why the evidence is sufficient. An argument that ignores stated assumptions, uncertainty, or contradictory evidence can create false confidence. And a trust decision that exceeds the supported trustworthiness creates risk by design.

[1] This is the rationale behind the use of the terminology *trustworthy secure*. Trustworthy is an essential modifier for the term secure and reflects the underlying realization that in a world of highly complex, interconnected systems, the concept of a *secure system* is extraordinarily difficult, if not impossible, to achieve. Trustworthiness is not absolute; it depends on the evidence associated with a specific claim and the assurance arguments made in support of that claim.

3. Trustworthiness Is a Continuum, Not a Binary Label

Trustworthiness is often discussed as though a system component or system is either trustworthy or untrustworthy. That binary framing is too coarse for engineering. SP 800-160 instead supports a continuum. The security design principle of *Commensurate Trustworthiness* states that a system element should be trustworthy to a level commensurate with the most significant adverse effect that can result from its failure [1]. The required level of trustworthiness therefore depends on *consequence*.

A system component whose failure produces minor inconvenience does not require the same evidentiary burden as a component whose compromise can cause loss of life, mission failure, disclosure of strategic intellectual property, loss of command authority, or cascading critical infrastructure disruption. The amount of trust placed in the component, the consequences if that trust is misplaced, and the evidence required to justify the trust must be considered together.

This leads to an important clarification. “Zero trust” as an architectural strategy should not be interpreted to mean zero trustworthiness or zero dependence. Real systems necessarily depend on system elements. The engineering objective is to avoid implicit, excessive, or unjustified trust; minimize the number of elements that must be trusted; bound the authority associated with each dependency; and demand evidence commensurate with the consequences of failure. At the other end of the continuum, “high assurance” should not be understood as perfect certainty. It represents a substantially stronger evidentiary basis, obtained through greater scope, depth, rigor, independence, and structured reasoning.

SP 800-160 reinforces this through the design principle of *Commensurate Rigor*. Rigor determines the scope, depth, and detail of an engineering activity. Generally, increased rigor increases confidence in the result, which can reduce uncertainty and risk. But rigor has cost and schedule consequences. The correct question is not whether every component should receive maximum assurance. It is where high assurance is necessary, where moderate assurance is sufficient, where low assurance can be tolerated because the architecture limits consequence, and where a component should not be trusted at all.

High assurance is not universal assurance. It is assurance concentrated where misplaced trust can produce the greatest loss.

---

4. The Assurance Case: Turning Evidence into a Defensible Trust Argument

The trustworthiness context in SP 800-160 is grounded in the *assurance case*. An assurance case is a reasoned, auditable artifact created to support the contention that a top-level claim is satisfied. It contains systematic argumentation, evidence, and explicit assumptions that support the claim [6]. In systems security engineering, the top-level claim may be that the system is adequately secure, that a critical security property is preserved, or that a specific architecture provides the protection required by stakeholder needs and objectives.

An assurance case is valuable because complex security properties cannot typically be demonstrated by a single test. Consider a claim that compromise of one administrative domain cannot result in unauthorized control of a mission-critical domain. Evidence may include architecture models, interface specifications, privilege analyses, configuration baselines, code or design reviews, penetration tests, verification results, supply chain evidence, operational monitoring data, recovery tests, and analyses of shared dependencies. The assurance case explains how those pieces collectively support the claim and where assumptions or uncertainty remain.

A useful assurance case structure includes:

- **Claims** — the security propositions that must be true, derived from stakeholder security objectives and protection needs.
- **Subclaims** — decompositions of the top-level claim into propositions that can be analyzed and evidenced.
- **Arguments** — the reasoning that explains why the evidence supports the claims and why the decomposition is valid.
- **Evidence** — objective results from analysis, demonstration, inspection, testing, evaluation, verification, validation, operational observation, and other credible methods.
- **Assumptions and constraints** — conditions that must hold for the argument to remain valid.
- **Residual uncertainty** — what is not known, cannot be known, or has not yet been substantiated. [6]

The assurance case also creates *traceability*. A stakeholder concern about unacceptable loss can be traced to security objectives, requirements, architecture, mechanisms, verification evidence, and the resulting confidence judgment. That traceability makes security review more than a checklist exercise. It enables reviewers to challenge the logic of the security argument, identify unsupported claims, locate hidden assumptions, and determine whether the evidence is commensurate with consequence.

SP 800-160 emphasizes that assurance cases must be *maintained* as the engineering effort changes. This is essential. A system update can invalidate a test result. A new connection can defeat a domain separation assumption. A supplier substitution can change the provenance of a trusted component. A recovery procedure can create a new privileged path. A new mission can increase the consequence associated with an old failure mode. Assurance that is not maintained becomes historical documentation rather than current evidence of trustworthiness.

5. Three Approaches to Assurance — and Why Assertion Is Not Enough

SP 800-160 describes three general approaches to assurance, from weakest to strongest: (1) axiomatic, (2) analytic, and (3) synthetic [1]. The distinction is especially useful because organizations frequently confuse compliance, testing, and assurance even though they provide different kinds of confidence.

Axiomatic assurance is assurance by assertion. Beliefs are accepted on faith in an artifact or process, often because nothing has yet contradicted them. Demonstrations of conformance and compliance can fall into this category when they are treated as the sole basis for confidence. Compliance has legitimate value: it can establish that prescribed activities were performed or specified requirements were addressed. But for complex systems, it is weak evidence that an emergent security property will hold under all relevant conditions.

Analytic assurance derives confidence from testing or reasoning about properties of interest. It moves the basis for belief from the artifact itself to the methods used to examine it. Testing, modeling, static analysis, formal analysis, fault analysis, penetration testing, modeling and simulation, and other techniques can materially strengthen confidence. But analytic assurance also has limits. Testing demonstrates the presence of defects more readily than their absence; models omit details; test environments differ from operational environments; assumptions can be wrong; and coverage can be incomplete. The resulting *uncertainty* must remain visible in the assurance argument.

Synthetic assurance is assurance by structured reasoning and composition. It treats assurance as something that must be considered throughout design and implementation, from constituent elements to the final system realization. The assurance case is an example of this approach. Synthetic assurance does not eliminate uncertainty, but it makes the reasoning explicit and uses the available evidence to reduce uncertainty in a disciplined way.

These approaches should not be interpreted as mutually exclusive. A mature assurance strategy can use conformance evidence, analytic evidence, and structured system-level reasoning together. The critical point is that the strength of the conclusion should not exceed the strength, relevance, and completeness of the evidence supporting it.

Compliance can tell us that required activities occurred. Assurance must tell us why we should believe the resulting system will protect what matters.

---

6. The Black-Box Problem: You Cannot Assure What You Cannot Examine

Modern systems increasingly depend on components whose internal construction, behavior, provenance, or failure modes are only partially visible to the system integrator. Commercial software, cloud services, proprietary hardware, machine-learning models, managed services, third-party libraries, firmware, and global supply chains can all create *black-box* dependencies. The component may provide a rich interface and impressive functionality while revealing little about the evidence needed to substantiate the security properties on which the system depends.

This creates an *assurance ceiling*. If systems engineers cannot inspect relevant design artifacts, evaluate development evidence, understand privileged behavior, characterize interfaces, test meaningful failure modes, establish provenance, or obtain credible independent evaluation, then some trustworthiness claims cannot be strongly substantiated. The problem is not that the system component is necessarily insecure. The problem is *epistemic*: the organization does not possess enough evidence to justify the level of trust it may be placing in the component.

Transparency therefore has engineering value. Transparency does not require that every implementation detail be public or that every user receive source code. It means that the parties responsible for making consequential trust decisions have access to evidence appropriate to the claims being made. Depending on the system component and consequence, that evidence may include architecture descriptions, interface specifications, secure-development records, vulnerability handling practices, test results, evaluation reports, provenance information, configuration guidance, formal models, software bills of materials, independent assessment results, or operational telemetry.

When transparency is constrained, the architecture should compensate. SP 800-160 design principles provide several options: (1) minimize the number of opaque elements that must be trusted; (2) reduce the privileges and functionality available to them; (3) separate them from higher-consequence domains; (4) mediate their interactions; (5) limit sharing; (6) introduce independent checks; (7) use redundancy or diversity where common-mode failure can be avoided; (8) monitor for anomalous behavior; and (9) design protective failure and recovery paths. In other words, uncertainty about a component should become an architectural input rather than an unspoken assumption.

This is particularly important for frontier AI models and components. A model can be powerful, adaptive, probabilistic, and difficult to specify exhaustively. Even extensive testing may not establish how it will behave across all prompts, contexts, tools, environmental conditions, or adversarial manipulations. The responsible conclusion is not that such a component can never be used. It is that its assurance may be bounded, and the architecture should limit the trust, authority, information access, and associated mission consequence assigned to it accordingly.

Opacity does not prove untrustworthiness. It limits the trustworthiness that can responsibly be claimed.

7. Assurance Gives the Security Design Principles Their Credibility

The 30 security design principles in SP 800-160 provide a powerful vocabulary for engineering trustworthy secure systems. But design principles alone do not create assurance. The engineering implementation of a principle must produce claims that can be substantiated. This is where the specific design principles and the assurance concepts become inseparable.

Domain Separation, for example, can support a claim that compromise or failure in one domain will not automatically propagate into another. But the label "separate" is not evidence. Assurance may require showing that communication paths are complete and known, mediation cannot be bypassed, management planes are independently protected, shared dependencies do not collapse the separation, failure modes preserve the intended boundary, and recovery does not reconnect domains in an unsafe state.

Least Privilege can support a claim that compromise yields bounded authority. Assurance requires evidence that privileges have been correctly identified, allocated, enforced, reviewed, and preserved across all system states, including exceptional states such as maintenance and recovery. *Minimal Trusted Elements* can reduce assurance cost by shrinking the set of elements whose failure can invalidate critical claims. *Reduced Complexity* can make evidence easier to obtain and arguments easier to reason about. *Structured Decomposition and Composition* makes it possible to organize assurance arguments at meaningful levels of abstraction.

Four principles are especially explicit about the assurance relationship. *Commensurate Rigor* aligns the depth of engineering activity with adverse consequence. *Commensurate Trustworthiness* aligns the required trustworthiness of a particular system element with the consequences of its failure. *Substantiated Trustworthiness* requires evidence rather than assumption. *Compositional Trustworthiness* recognizes that system-level trustworthiness is an emergent property and cannot be inferred simply from claims about individual system elements [1].

Together, these principles create a disciplined *trust strategy*: minimize what must be trusted; determine how trustworthy each trusted element must be; apply rigor commensurate with consequence; obtain evidence sufficient to substantiate the required properties; and separately establish that the composition preserves those properties at the system level.

8. Assurance: A Unifying Concept Across Trustworthy Secure Systems Engineering

Assurance provides a common evidentiary foundation for several systems security engineering ideas that are often discussed separately. Viewed through the trustworthiness lens in SP 800-160, they become parts of one argument about justified confidence.

Trustworthy Secure Design. Designing security into architecture before selecting controls reduces the number and complexity of claims that later must be substantiated. Eliminating an attack path is generally easier to assure than depending on a complex mechanism to detect and block every possible exploitation of that path. The *Security Design Order of Precedence* therefore has an assurance benefit as well as a security benefit: earlier design choices can simplify the evidentiary burden. [7]

Adequate Security. The judgment that a system is adequately secure is inherently an assurance judgment. Minimum tolerable security and "as secure as reasonably practicable" cannot be established by intuition alone. Stakeholders need evidence that the system's security performance, including its behavior, is sufficient for the consequences at stake and that additional improvements have been considered in the engineering trade space. Adequacy without assurance is simply a declaration. [8]

Engineering for Compromise. Claims about bounded compromise, containment boundaries, propagation barriers, compromise margin, protective failure, and recovery all depend on evidence. A boundary that has never been tested against bypass paths, shared authorities, maintenance states, or common dependencies is an architectural aspiration, not an assured containment property. EfC becomes credible when the causal chain from component compromise to unacceptable loss is accompanied by evidence that the intended barriers actually hold. [9]

Trustworthy Parts and Composition. Component evidence provides the starting point for composition, not the conclusion. High-assurance components can be integrated into a low-assurance system if the interfaces, dependencies, configurations, or emergent behaviors invalidate their individual properties. Conversely, components with bounded assurance may sometimes be used safely if the architecture constrains the consequences of their failure. The system assurance case must explain both the trustworthiness of critical system components and the trustworthiness of their composition. [10]

Multidimensional Protection and Resilience. Layered protection strategies require evidence that the layers are effective, sufficiently independent, and collectively address the relevant loss scenarios. Resilience requires assurance that the system can withstand, recover, and adapt without restoring to a compromised state or creating new loss paths. Monitoring contributes evidence during operation, but operational evidence must be connected back to the claims it is intended to support. [11]

Assurance is the thread that turns design principles, resilience mechanisms, and security architectures into justified claims of trustworthiness.

---

9. Assurance Must Be Engineered Across the System Life Cycle

Assurance is not a test phase at the end of development. SP 800-160 describes assurance as a complex, multidimensional property that builds over time and must be planned, established, and maintained throughout the system life cycle [1]. This changes how evidence should be treated. Evidence is not a documentation by product; it is a systems engineering *deliverable*.

During the *mission and stakeholder analysis* stages of the system life cycle, assurance begins by making protection needs and unacceptable consequences explicit enough to support meaningful claims. During requirements engineering, claims become testable expectations and measures of success. Architecture and design determine where trust is placed, which elements require higher assurance, and which structural decisions can reduce the evidentiary burden. Implementation produces evidence about realization. Integration exposes compositional assumptions. Verification and validation determine whether specified

requirements are satisfied and whether the resulting system meets stakeholder needs in its intended environment.

Transition and operation add evidence that cannot be fully generated in development: deployment configuration, real workload behavior, operator interaction, environmental conditions, attack telemetry, failure patterns, and recovery performance. Maintenance can strengthen or invalidate earlier assurance. Disposal introduces its own claims about sanitization, residual information, decommissioned credentials, and removal of trust relationships.

The assurance case should therefore evolve with the system. When a security-relevant change occurs, the question should not be limited to whether the change works. Engineers should ask which claims depend on the changed element, which evidence is now outdated, which assumptions no longer hold, whether the composition argument remains valid, and what new evidence is needed to restore justified confidence.

This holistic, system life cycle view also addresses a common weakness in system authorization approaches. A one-time assessment can establish confidence about a particular system realization at a particular time. It cannot guarantee that confidence survives years of software updates, cloud migrations, supplier changes, new missions, altered threat conditions, or accumulated operational workarounds. Continuous change requires continuously maintained assurance.

10. Designing an Assurance Strategy Commensurate with Consequence

Not every claim warrants the same evidentiary investment. Assurance itself belongs in the engineering trade space. SP 800-160 recognizes that the form of an assurance case and the rigor and formality used to acquire evidence should reflect the target level of assurance, the consequences involved, and the size and complexity of the system [1]. The goal is disciplined sufficiency, not evidence accumulation for its own sake.

A practical assurance strategy can begin with five questions:

1. What are the most consequential security claims? Identify the claims whose failure would expose stakeholders to unacceptable loss.
2. Which system elements and interactions are trusted for those claims? Map the dependencies that can invalidate each claim.
3. How much assurance is required? Scale rigor to consequence, uncertainty, complexity, exposure, and the significance of misplaced trust.
4. What evidence can actually be obtained? Determine whether the necessary evidence is available, credible, independent where needed, and relevant to the claim.
5. What must change when assurance is insufficient? Reduce trust, redesign the architecture, add independent barriers, constrain functionality, obtain stronger evidence, or explicitly accept the residual risk.

This approach makes assurance *actionable*. It prevents organizations from treating every component as equally important and every piece of evidence as equally valuable. It also exposes where procurement and acquisition decisions create unavoidable assurance gaps. If a supplier cannot provide evidence needed to justify a critical trust dependency, the organization has three choices: obtain stronger evidence through another mechanism, architect around the uncertainty, or knowingly accept the risk. What it should not do is silently convert lack of evidence into confidence.

The same logic applies to emerging technologies. Novelty does not reduce the need for assurance; it may increase it because uncertainty is higher. At the same time, traditional assurance methods may not map cleanly to new components. The response should be to adapt the evidentiary strategy while preserving the core principle: **claims of trustworthiness must remain proportional to what the evidence can support**. Figure 1 shows the entities that contribute to the rigor of systems and resulting degrees of assurance obtained.

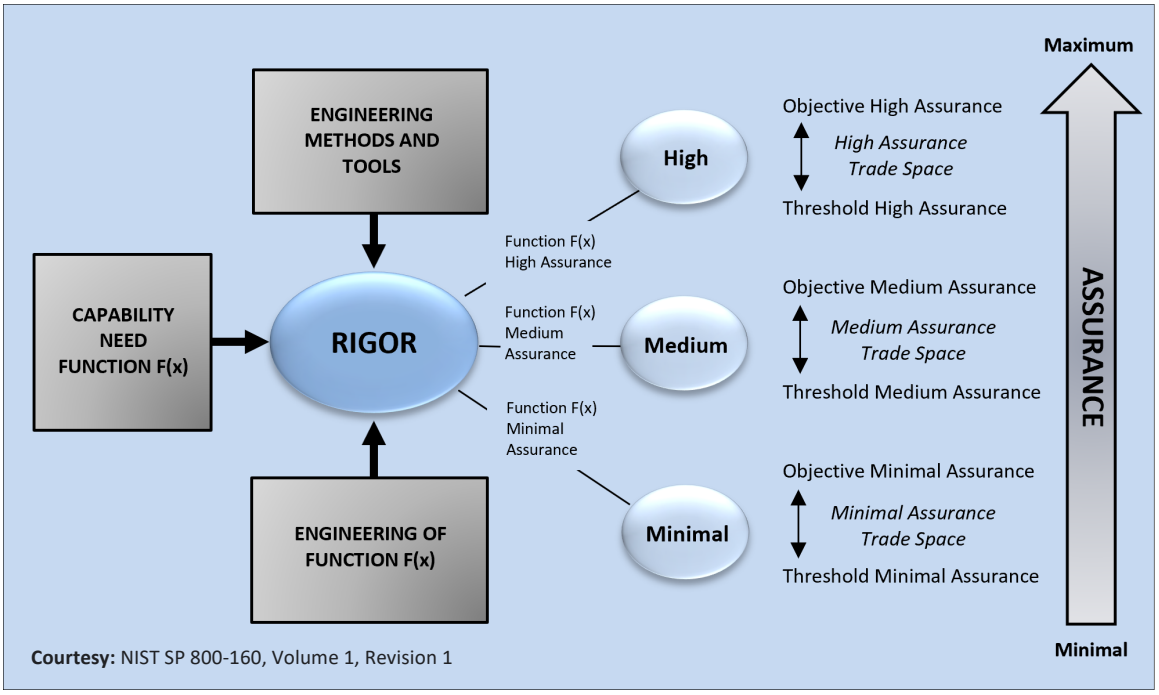

Figure 1: Assurance and Degree of Rigor in Realizing a Capability Need

11. A Practical Assurance Review for Engineering and Executive Teams

Assurance should be understandable not only to evaluators and system security engineers but also to mission owners, acquisition officials, executives, and authorization decision-makers. A concise review can expose whether a security argument is genuinely evidence-based or rests primarily on confidence by assertion.

- What are we claiming about this system that must remain true under adversity?
- What unacceptable losses depend on those claims?
- Which components, people, services, suppliers, and enabling systems are trusted for each claim?
- What objective evidence supports the trustworthiness of those dependencies?
- Is the evidence relevant to the actual configuration, mission, operating environment, and system state?
- What is the level of rigor behind the evidence, and is that rigor commensurate with consequence?
- What assumptions are embedded in the argument, and how are violations of those assumptions detected?
- Where are the black boxes, and what assurance ceiling do they impose?
- Does the composition preserve the properties attributed to the individual parts?

- What evidence supports containment, failure behavior, and recovery, not merely nominal operation?
- What has changed since the evidence was generated?
- What uncertainty remains, and who has explicitly accepted the risk associated with it?

These questions shift executive discussion away from the quantity of security controls and toward the *quality* of the security argument. They also make uncertainty visible. That is a strength, not a weakness. Mature engineering does not require pretending that uncertainty has disappeared. It requires knowing where uncertainty remains, reducing it where the consequences justify the effort, and constraining the system so that residual uncertainty does not create disproportionate risk.

12. Conclusion: Trust Must Be Earned by Evidence

Trustworthy secure systems engineering ultimately depends on a simple discipline: do not confuse what the system is intended to do with what there is reason to believe it will actually do. Architecture establishes the intended security properties. Design principles help realize them. Verification, validation, analysis, testing, evaluation, operational observation, and other methods produce evidence. The assurance case organizes that evidence into a reasoned argument. Assurance provides justified confidence. That confidence supports a trustworthiness judgment. Only then is there an *engineering basis* for granting trust.

This is why assurance is the unifying concept across trustworthy secure design. It connects the problem context to the solution context and then asks whether the resulting system has earned the confidence stakeholders are being asked to place in it. It connects component trustworthiness to compositional trustworthiness. It connects adequate security to evidence. It connects compromise containment to demonstrated boundaries. It connects resilience to trustworthy recovery. It connects design principles to proof that their intended properties survive implementation, integration, operation, and change.

The implications for black-box systems are equally direct. Limited visibility does not automatically make a component untrustworthy, but it limits the evidence available to substantiate its claims. That limitation must constrain the amount of trust placed in the component or motivate architectural measures that bound the consequences if the trust is misplaced. Trustworthiness is not a marketing label. It is a conclusion whose strength should rise and fall with the strength of the evidence.

In a world of increasingly complex supply chains, cloud dependencies, autonomous software, opaque services, and frontier AI, assurance becomes more, not less, important. The more difficult it is to know what is inside the system, the more disciplined engineers must be about what they claim to know about its behavior. The objective is not certainty. It is justified confidence commensurate with consequence.

If trust is a decision to depend, assurance is the evidence that tells us whether that dependence is justified.

“The trust we place in our digital infrastructure should be proportional to how trustworthy and transparent that infrastructure is and to the consequences we will incur if that trust is misplaced.”

-- Executive Order (EO) on Improving the Nation’s Cybersecurity [12]

May 2021

Acknowledgments

The author acknowledges the Institute for Security, Technology, and Society (ISTS) at Dartmouth College for supporting 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 or the National Institute of Standards and Technology.

References

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

[2] Schroeder M, Clark D, and Saltzer J (1977) "The Multics Kernel Design Project." Sixth ACM Symposium on Operating Systems Principles. https://web.mit.edu/Saltzer/www/publications/rfc/csr-rfc-140.pdf

[3] Levin T, Irvine C, Benzel T, Bhaskara G, Clark P, and Nguyen T (2007) "Design Principles and Guidelines for Security," Technical Report NPS-CS-07-014. Naval Postgraduate School. https://nps.edu/web/c3o/technical-reports

[4] Neumann P (2004) "Principled Assuredly Trustworthy Composable Architectures," CDRL A001 Final Report, SRI International, Menlo Park, CA. http://www.csl.sri.com/users/neumann/chats4.pdf

[5] International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers (2019) ISO/IEC/IEEE 15026-1:2019, *Systems and software engineering – Systems and software assurance – Part 1: Concepts and vocabulary*. https://www.iso.org/standard/73567.html

[6] International Organization for Standardization/International Electrotechnical Commission (2011) ISO/IEC 15026-2:2011 – *Systems and software engineering – Systems and software assurance – Part 2: Assurance case*. https://www.iso.org/standard/80625.html

[7] Ross R (2026) "Trustworthy Secure Design: Engineering Security from the Inside Out," LinkedIn, August 2026.

[8] Ross R (2026) "Adequate Security: An Engineering Answer to an Impossible Question," LinkedIn, August 2026.

[9] Ross R (2026) "Engineering for Compromise: Designing Systems That Remain Trustworthy Secure Under Adversity," LinkedIn, August 2026.

[10] Ross R (2026) "Trustworthy Systems Need Trustworthy Parts," LinkedIn, August 2026.

[11] Ross R (2026) "A Multidimensional Protection Strategy for Engineering Mission-Resilient Systems in the Age of Frontier AI," LinkedIn, July 2026.

[12] Executive Order 14028 (2021), Improving the Nation's Cybersecurity. (The White House, Washington, DC), May 12, 2021. https://www.federalregister.gov/d/2021-10460