

The Art of Systems Security

What Sun Tzu Can Teach Us About Engineering Trustworthy Secure Systems

Dr. Ron Ross

CEO, RONROSSECURE, LLC
ISTS Fellow, Dartmouth College

Abstract

Sun Tzu's The Art of War and NIST Special Publication 800-160, Volume 1 arise from radically different eras, purposes, and disciplines. Yet both place extraordinary value on preparation, knowledge, positioning, disciplined control, adaptation, and the deliberate shaping of conditions before adversity is encountered. This paper uses Sun Tzu's strategic ideas as an interpretive lens for the 30 principles for trustworthy secure design in NIST SP 800-160. It does not claim historical lineage, direct equivalence, or a one-to-one mapping between military doctrine and systems engineering. Instead, it shows how ancient strategic insights can make modern engineering ideas more accessible: understand the mission and environment; reduce uncertainty; shape the architectural terrain; deny an adversary concentrated advantage; preserve control when conditions change; and substantiate claims with evidence. The central argument is that trustworthy secure systems are not produced by accumulating controls after design decisions have been made. They are engineered by arranging trust, authority, information flow, interfaces, dependencies, failure behavior, and recovery so that unacceptable loss is made difficult to reach. Sun Tzu sought advantage before battle. Systems security engineering seeks protection before compromise.

1. Introduction

Cybersecurity is often described as a contest of tools, exploits, vulnerabilities, and defenses. That description is incomplete. Before any technical exchange occurs, the architecture has already allocated advantage. It has decided where trust is concentrated, which system elements can act unilaterally, what information is shared, which dependencies cross boundaries, how failures propagate, and whether recovery can restore a trustworthy secure state. Those decisions determine what an adversary inherits after a successful compromise.

Sun Tzu's enduring contribution is not a catalogue of battlefield maneuvers. It is a way of thinking about conflict as a problem of knowledge, preparation, positioning, economy, and control. The best result is achieved by shaping conditions so thoroughly that the adversary is denied a favorable path. NIST SP 800-160 offers a modern engineering counterpart: principles for reasoning about system trustworthiness and controlling the effects of adversity across the life cycle [1].

2. A Strategic Lens—Not a Claimed Equivalence

The Art of War is a military treatise. NIST SP 800-160 is a systems security engineering publication. The former addresses commanders, forces, terrain, intelligence, timing, and conflict; the latter addresses systems, stakeholders, assets, loss, requirements, architecture, design, life-cycle processes, and evidence. Collapsing those domains into literal equivalence would distort both.

The useful relationship is at the level of *strategic reasoning*. Sun Tzu repeatedly directs attention away from isolated engagements and toward the conditions that make outcomes possible. He emphasizes knowing oneself and the adversary, understanding terrain, avoiding strength, exploiting weakness, preserving

resources, maintaining discipline, and changing methods as circumstances change [2]. Systems security engineering asks corresponding—not identical—questions. What does the mission depend upon? What losses are unacceptable? Which system states can produce them? Where are trust and authority placed? What can a compromised element reach? Which boundaries, constraints, and response capabilities interrupt the causal path to loss?

Sun Tzu sought to deny an adversary favorable conditions. Systems security engineering seeks to deny a compromised element the conditions necessary to produce unacceptable loss.

This lens clarifies why security is an *emergent* property of an engineered system. A strong component cannot rescue an architecture that grants excessive authority, permits uncontrolled propagation, or depends on an untrustworthy recovery path. Conversely, a system can remain trustworthy secure under adversity when its architecture limits what any one failure or compromise can accomplish. The relevant unit of reasoning is therefore the system in its *environment*—not the security feature in *isolation*.

NIST cautions that its principles are not compliance rules, are not prescriptively ordered, and may conflict or require trade-offs in a particular context [1]. The six groupings that follow are explanatory constructs. They organize all 30 principles around recurring strategic ideas in *The Art of War* without changing the meaning or scope of the principles themselves.

3. Win Before the Engagement

Sun Tzu’s strategist seeks battle only after creating the specific conditions for *success*. The systems security engineering analogue is to establish a defensible system concept, architecture, and body of evidence before an operational adversary tests them. Five NIST security design principles are especially important to that preparation.

- **Clear Abstractions** — Represent the system with models, interfaces, and concepts that are simple, precise, necessary, and sufficient. Strategic preparation fails when decision-makers cannot see the real system or reason from an abstraction that hides a consequential dependency.
- **Commensurate Rigor** — Match the depth, scope, and formality of engineering activity to the most significant adverse effect that can occur. The effort used to analyze a non-essential function should not determine the assurance applied to a function whose failure can produce catastrophic loss.
- **Substantiated Trustworthiness** — Base judgments of trustworthiness on evidence, not reputation, assertion, labels, or optimistic assumptions. Claims should identify assumptions, scope, evidence, residual uncertainty, and the conditions that require reassessment.
- **Loss Margins** — Maintain sufficient distance between the system’s normal or degraded operating state and a state that produces unacceptable loss. Margin gives defenders time and options. A system continually at the edge of loss has surrendered strategic depth.
- **Commensurate Protection** — Provide each system element with protection proportional to the most significant adverse effect of its failure. Elements carrying greater mission trust require stronger protection and stronger confidence (i.e., greater assurance) in that protection.

The strongest security position is not created during the attack. It is established in the requirements, design, architecture, and evidence before the attack begins.

The interaction of these principles establishes a justified security position before the system enters operation. *Clear Abstractions* and *Loss Margins* define what must be understood and how much distance the system must maintain from unacceptable loss. *Commensurate Protection* and *Commensurate Rigor* align protection and assurance effort with consequence, while *Substantiated Trustworthiness* tests whether the resulting claims deserve confidence. The strategic result is prepared advantage: critical assumptions are exposed, loss-control logic is designed into the architecture, and consequential claims are supported before adversity occurs.

4. Know the Adversary—and Know the System

Sun Tzu’s best-known strategic admonition joins knowledge of the adversary with knowledge of oneself. For engineered systems, self-knowledge must be continuous. The system must be understandable, observable, and stripped of avoidable exposure.

- **Anomaly Detection** — Detect salient deviations in behavior, outcomes, state, configuration, or environment soon enough to enable an effective response. Detection requires a model of expected behavior, trustworthy data collection, and interpretation related to possible loss, not merely a volume of alerts.
- **Minimize Detectability** — Reduce unnecessary signatures, interfaces, access points, emissions, responses, and other exposures that help an adversary discover, observe, track, or target the system. What cannot practicably be hidden must be deliberately protected and monitored.
- **Least Persistence** — Keep functions, resources, data, connectivity, privilege, and system elements present or usable only for as long as needed. Removing or disabling what is not required reduces the time and capability available for misuse, compromise, or unauthorized activation.
- **Reduced Complexity** — Make the design as simple as practicable. Unnecessary complexity conceals dependencies, multiplies ambiguous states, frustrates verification, and creates places where adversary action can remain unnoticed. Simplicity is a means of reducing *uncertainty*.
- **Structured Decomposition and Composition** — Divide the system into coherent elements with well-defined responsibilities, interfaces, layers, and dependencies, then compose those elements so their interactions preserve system-level requirements. Decomposition makes the terrain understandable; disciplined composition prevents local behavior from defeating global intent.

These principles turn *situational awareness* inward and outward. *Minimize Detectability* concerns what the adversary can learn about the system. *Anomaly Detection* concerns what the system and its operators can learn about changing conditions. *Least Persistence* removes stale opportunities. *Reduced Complexity* and *Structured Decomposition and Composition* make the system sufficiently intelligible to analyze and control.

An architecture that cannot be understood cannot be confidently defended; a system that cannot observe itself cannot reliably know when its security argument is failing.

The implication is practical. An inventory without dependency knowledge is incomplete. Telemetry without a behavioral model is noise. A diagram that omits administrative, supply-chain, recovery, identity, cloud, physical, or human dependencies is not a trustworthy abstraction of the operational system. Knowing the system means knowing what it depends upon, what it exposes, what it can observe, and how those facts change across normal, degraded, maintenance, emergency, and recovery modes.

5. Shape the Architectural Terrain

Sun Tzu treated terrain as a decisive factor in shaping military advantage. It enables some movements, constrains others, concentrates power, creates delay, exposes weakness, and determines whether forces can support one another. Architecture plays a comparable strategic role in a system. It establishes the paths through which information, authority, control, failure, and compromise can move.

- **Domain Separation** — Partition the system into domains with distinct policies, trust assumptions, information, authority, or consequence. Effective separation must address more than network topology; shared identities, administrators, management planes, storage, updates, and recovery mechanisms can silently reconnect domains.
- **Least Privilege** — Grant each entity only the authority needed to perform its assigned function under defined conditions. Because a compromised entity can exercise its legitimate privileges maliciously, every unnecessary privilege is latent adversary capability.
- **Least Sharing** — Minimize shared mechanisms, resources, services, data, and dependencies. Sharing can improve efficiency, but it also creates coupling and common paths for interference or propagation. The security cost of sharing across consequential boundaries should be explicit and justified.
- **Least Functionality** — Provide only the functions required to satisfy stakeholder needs. Unnecessary protocols, connections, services, interpreters, interfaces, modes, and administrative features expand both attack opportunity and the capability inherited through compromise.
- **Mediated Access** — Ensure that access to protected resources and consequential interactions is controlled by mechanisms that are always invoked, tamper resistant, non-bypassable, and analyzable. Mediation may need to constrain action, direction, state, volume, timing, and context as well as identity.

Applied in concert, the principles convert architecture into controlled terrain. *Domain Separation* establishes where compromise should stop; *Least Privilege*, *Least Sharing*, and *Least Functionality* reduce what an adversary can acquire within and across those boundaries; and *Mediated Access* governs the interactions the mission still requires. Their combined effect is to turn successful entry into bounded access rather than unrestricted system-wide reach.

This is where architecture converts prevention into *containment*. A vulnerability may permit entry into one domain, but the terrain should deny automatic access to every other domain. A credential may be stolen, but it should not authorize every consequential action. Good architectural terrain makes an adversary spend additional time, expose additional behavior, and defeat independent constraints.

6. Make One Success Insufficient

Sun Tzu warns against allowing an adversary to apply strength where one is weak. Modern systems frequently create exactly that opportunity by concentrating trust in a single administrator, control plane, credential, component, supplier, or protective mechanism. The following principles distribute and diversify the means by which the system prevents loss.

- **Distributed Privilege** — Require independent entities, authorities, or conditions to cooperate before selected consequential actions can occur. Threshold cryptography, split administrative domains, two-person control, and separate approval chains can ensure that one compromised element is insufficient.
- **Defense in Depth** — Provide multiple coordinated lines of protection so that no single mechanism bears the entire loss-control burden. The lines must be meaningfully independent and should not conceal dangerous conditions or share a common mode of defeat.

- **Diversity (Dynamicity)** — Use different implementations, mechanisms, configurations, locations, or changing system characteristics to reduce susceptibility to common-mode failure and predictable attack. Diversity that merely adds complexity without reducing correlated failure does not create strategic advantage.
- **Redundancy** — Replicate functions, elements, paths, or data to avoid single points of failure and sustain required capability. Redundancy improves availability only when failure detection, selection, synchronization, and common-mode risks are addressed.
- **Minimal Trusted Elements** — Minimize the set of system elements whose failure or malicious behavior can invalidate critical security claims. The smaller and simpler the trusted set, the more feasible it becomes to apply commensurate rigor, protection, analysis, and evidence.

A resilient architecture does not ask one component to be perfect. It arranges the system so that one component's failure is not decisive.

The five principles are complementary but not interchangeable. *Redundancy* without *Diversity* may reproduce the same defect. *Defense in Depth* without independence may create several visible layers supported by one hidden dependency. *Distributed Privilege* without trustworthy separation may place multiple approvals under one administrator. *Minimal Trusted Elements* does not eliminate trust; it exposes and concentrates assurance effort on the smallest practicable set of consequential dependencies.

The strategic result is *asymmetry*. The defender uses architecture to require an adversary to defeat multiple independent conditions while legitimate users continue to perform the mission. Every additional condition must be purposeful and analyzable; otherwise, added layers can create complexity that can benefit the adversary more than the defender.

7. Retain Control Under Adversity

For Sun Tzu, preparation and positioning mattered only if a commander could maintain disciplined control as conditions changed and adversity intensified. The systems security engineering analogue is a system that continues to enforce its security intent when assumptions fail. This requires mechanisms that mediate consequential behavior, preserve protection across system states and modes, constrain the effects of failure, and enable trustworthy recovery.

- **Trustworthy System Control** — Design constraint-enforcing functions to be tamper resistant, always invoked, non-bypassable, and sufficiently small and simple to support rigorous analysis and testing. The elements being controlled must not be able to disable or redefine the controls that constrain them.
- **Continuous Protection** — Maintain the required protection throughout every system state, mode, and life-cycle transition. Startup, shutdown, maintenance, update, degraded operation, emergency access, recovery, and disposal are part of the security argument, not exceptions to it.
- **Protective Defaults** — Configure the system so its default state favors loss prevention and protection effectiveness. Capability that requires reduced protection should result from an authorized, deliberate, and observable decision rather than an insecure initial condition or silent fallback.
- **Protective Failure** — Constrain a system-element failure so it neither causes unacceptable loss nor activates a different loss scenario. Failure behavior must be engineered, including the possibility that attempts to preserve service can worsen the security condition.

- **Protective Recovery** — Restore functions, assets, and services without advancing the original loss scenario, creating a new one, or relying on a compromised basis of trust. Recovery should reestablish an authenticated, authorized, known, and sufficiently trustworthy state.

The coordinated application of the principles forms a control structure that persists through adversity. *Trustworthy System Control* and *Continuous Protection* maintain the enforcement of critical constraints, while *Protective Defaults*, *Protective Failure*, and *Protective Recovery* govern the system's starting condition, its behavior during disruption, and its return to an acceptable state. None is sufficient alone: control that disappears during failure or recovery cannot preserve the system's security intent.

The strategic result is *continuity* of security intent. Systems are often best protected during normal operation but weakest when conditions become abnormal. Emergency accounts can accumulate privilege. Maintenance channels can bypass production controls. Monitoring can disappear during startup. Recovery images can become stale, and availability mechanisms can fail open. The architecture must therefore preserve critical security invariants across those transitions and specify what may change, who may authorize the change, what evidence must be retained, and how the system demonstrates that recovery has restored, not merely restarted, trustworthy secure operation.

8. Respond, Adapt, and Preserve the Whole

Sun Tzu rejects rigid methods detached from circumstances. Effective action depends on conditions, consequence, timing, and the behavior of the whole force. Systems security engineering expresses that adaptability through proportional response, layered trust relationships, and explicit reasoning about how local properties compose into system-level trustworthiness.

- **Commensurate Response** — Match the aggressiveness and timing of a response to the immediacy and consequence of the loss scenario, including the response's own potential adverse effects. Early detection preserves the option for graduated action; imminent catastrophic loss may require an immediate ungraduated response.
- **Commensurate Trustworthiness** — Ensure that the trustworthiness of a system element is proportional to the most significant adverse effect of its non-performance. Trust assigned by the architecture must never exceed the evidence-supported worthiness of the element to bear that trust.
- **Hierarchical Protection** — Organize protection so that elements at higher levels of privilege, criticality, or abstraction are protected from and can reliably constrain elements below them. A lower-trust element must not be able to alter the mechanism or evidence on which higher-level control depends.
- **Compositional Trustworthiness** — Establish that the composition of elements preserves required system-level trustworthiness properties. Secure components do not automatically make a secure system; interfaces, emergent behavior, shared dependencies, and interactions can invalidate local assurances.
- **Self-Reliant Trustworthiness** — Achieve an element's trustworthiness with minimal dependence on other elements. When external protection is unavoidable, those dependencies and their trustworthiness must be explicit because degradation of the protecting element can invalidate the original claim.

Trustworthiness is not inherited from the strongest component. It emerges from the way all consequential elements, dependencies, and controls work together.

The combination of principles governs how trust is allocated, composed, and preserved as conditions change. *Commensurate Response* and *Commensurate Trustworthiness* align protective action and assigned trust with potential consequences. *Hierarchical Protection*, *Compositional Trustworthiness*, and *Self-Reliant Trustworthiness* prevent weaker elements, interactions, or external dependencies from silently invalidating system-level claims. Their combined effect is to preserve justified trust without assuming that trustworthy components automatically produce a trustworthy system.

The strategic result is *adaptive trustworthiness*. The system can respond to changing conditions without abandoning its security intent or assuming that trustworthy system components automatically produce a trustworthy system. A cryptographic module may protect keys yet depend on untrustworthy provisioning. A resilient service may depend on a fragile identity provider. A well-tested application may be deployed through a compromised pipeline. The architecture must therefore make trust dependencies explicit, evaluate how local properties compose, and ensure that protective responses remain proportional to the loss being controlled. Trade-offs remain unavoidable, but they become visible, analyzable, and supported by evidence rather than hidden within the system's complexity.

9. From Strategic Insight to Engineering Practice

The value of an analogy is measured by the action it enables. The following sequence turns the strategic lens into a practical *systems security engineering* activity. It is not a replacement for the life-cycle processes in NIST SP 800-160. It is a concise way to use the 30 principles to interrogate architecture and develop an evidence-based security argument.

1. **Define the mission and unacceptable losses** — Identify the assets, capabilities, and consequences that matter to stakeholders. State what must be prevented regardless of the threat, vulnerability, fault, error, or event sequence that might produce the loss.
2. **Model the operational system** — Include enabling systems, users, the supply chain, cloud services, networks, identities, administrative paths, maintenance functions, physical interfaces, and recovery infrastructure. Examine normal, degraded, contingency, emergency, and transition states.
3. **Identify hazardous conditions and causal paths** — Determine which system states, control actions, information flows, or failures can lead to each unacceptable loss. Consider malicious control of system elements as well as unintentional non-performance and correlated failure.
4. **Map trust and authority** — For every important claim, identify which system elements can invalidate it. Ask what a fully compromised element could read, alter, authorize, suppress, impersonate, invoke, or influence using all legitimate access available to it.
5. **Shape the architecture** — Apply separation, minimization, mediation, distributed privilege, defense in depth, diversity, redundancy, protective failure, and trustworthy control to interrupt causal paths and preserve loss margins.
6. **Define observable invariants and responses** — Specify what must remain true, what evidence reveals deviation, how quickly detection must occur, who or what may respond, and how the aggressiveness of the response relates to consequence and available time.
7. **Engineer protective recovery** — Define the trustworthy basis, authorization, provenance, integrity checks, reconstitution steps, validation criteria, and evidence required to return the system to an acceptable state without reopening the loss path.
8. **Construct and maintain an assurance case** — Organize claims, arguments, evidence, assumptions, residual uncertainty, and scope. Tie evidence to the as-built and as-operated system, and revisit the case when architecture, implementation, environment, dependencies, mission, or threat conditions change.

10. Assurance Is the Strategic Reserve

Sun Tzu's strategist does not rely on hope. Systems security engineering should not rely on unsupported security claims. An *assurance case* makes the reasoning visible: what is claimed, why evidence supports it, where the claim applies, which assumptions it depends upon, what could defeat it, and how uncertainty is managed. It can incorporate architecture models, requirements traceability, hazard and compromise analyses, interface specifications, code and configuration evidence, test results, formal analysis, operational telemetry, provenance records, incident evidence, and independent evaluation.

The assurance case is also a life-cycle instrument. When a system component, integration, administrative capability, AI function, supplier, operational environment, or recovery method changes, the assurance case identifies which claims and evidence must be revisited. In that sense, assurance provides *strategic reserve*: it preserves justified confidence while the system and its environment evolve.

11. Where the Analogy Ends

The analogy between Sun Tzu and systems security engineering has boundaries. A system is not an army, a user is not an enemy, and ordinary engineering failure is not combat. Security must account for safety, privacy, reliability, usability, law, ethics, cost, and mission needs that cannot be reduced to competitive advantage. Military language can also encourage an overly threat-centric mindset when many unacceptable losses arise from error, defect, accident, environmental stress, or poorly understood interaction.

Nor should memorable strategic maxims substitute for rigorous analysis. The 30 principles in NIST SP 800-160 have precise engineering meanings and documented relationships. They are contextual, can conflict, and require evidence-based judgment. Sun Tzu can sharpen the questions engineers ask, but he cannot answer whether a particular architecture satisfies stakeholder protection needs. That conclusion must come from the system's requirements, models, implementation, operational evidence, and assurance argument.

12. Conclusion

The Art of War endures because it directs attention to the structure of advantage before conflict begins. NIST SP 800-160 directs the same quality of attention, through a modern engineering discipline, to the structure of trustworthiness before adversity occurs. Both perspectives reject the idea that success is achieved solely through *reactive strength* at the moment of engagement.

For systems security engineering, the decisive work occurs when stakeholders define unacceptable loss; engineers model the real system and its environment; architects place boundaries around trust and authority; designers minimize and mediate consequential interactions; developers preserve intended constraints; operators detect changing conditions; and evidence supports claims throughout the system life cycle. The 30 principles provide a disciplined vocabulary for that work.

The art of systems security is to shape the architecture so that adversity can occur without acquiring a path to unacceptable loss.

Sun Tzu teaches us to prepare, know, position, constrain, preserve, and adapt. NIST SP 800-160 shows how those strategic instincts can be transformed into *trustworthy secure design*. The result is not invulnerability. It is a system engineered to deny easy advantage, to remain under trustworthy control, to fail and recover protectively, and to provide evidence worthy of stakeholder confidence.

Acknowledgments

The author gratefully acknowledges the support provided by Dartmouth College's Institute for Security, Technology, and Society (ISTS). The resources provided by the ISTS contributed to the research and development of this paper. The views and conclusions expressed herein are those of the author and do not necessarily represent the views of Dartmouth College.

References

- [1] R. Ross, M. Winstead, and M. McEvilly, *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] Sun Tzu, *The Art of War*, translated by Lionel Giles, 1910, Project Gutenberg eBook No. 132. <https://www.gutenberg.org/ebooks/132>