Examination of Security Design Principles from NIST SP 800-160

Logan O. Mailloux*, Paul M. Beach, and Martin "Trae" Span
Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio
*Corresponding Author: Logan.Mailloux@afit.edu

Abstract—**This paper explores the engineering of secure and resilient systems through a detailed examination of security strategies and principles as presented in Appendix F of the recently published National Institute of Standards and Technology Special Publication (NIST SP) 800-160. First, a brief introduction to systems security engineering is provided with recommended readings for those who desire to become more familiar with the specialty domain. Next, the NIST SP 800-160 Appendix F systems security strategies and principles are described, as well as, examined for implementation considerations. This examination and mapping provides a linkage of abstract security strategies to concrete security principles which can be more directly implemented, traced, and tested.**

Keywords—Systems Security Engineering; Systems Engineering; Security Principles; Design Principles

I. INTRODUCTION

Modern systems are increasingly complex with several subsystems, supporting & enabling systems, and extensive infrastructure dependencies which result in interactive and emergent behaviors. As unprecedented system-of-systems, they are inherently susceptible to a wide range of malicious and non-malicious events which can result in unexpected disruptions and unpredictable security related behaviors. This systems security gap arises from a lack of rigorously applied security analysis and engineering [1]. Thus, special attention is required to engineer secure and resilient systems built to operate in highly contested operational environments fraught with uncertainty, unpredictability, and attacks from intelligent adversaries, as well as, abuse and misuse by humans (e.g., owners, operators, maintainers, etc.) [2], [3], [4].

To address this critical systems security gap, the National Institute of Standards and Technology (NIST), National Security Agency (NSA), MITRE, and several industry leaders from around the world collaborated on a five-year effort to produce a comprehensive Systems Security Engineering (SSE) approach [5]. In November of 2016, the final version of NIST Special Publication (SP) 800-160, *Systems Security Engineering*, was released with the goal of formalizing and institutionalizing engineering-driven actions to develop more defensible and resilient systems [1]. This work is part of an ongoing research activity to raise awareness regarding the revitalization of SSE with an emphasis on more fully understanding its tailored application to assist developers, owners, and operators in understanding and achieving a rigorous SSE approach [6], [7].

This paper provides a straightforward introduction to SSE while highlighting essential definitions and concepts. Sections III and IV describe the NIST SP 800-160 Appendix F security developmental strategies and maps them to 18 architectural and design principles. In addition, these mappings are examined for implementation considerations and tradeoffs. This examination is useful for meeting system security needs by mapping conceptual strategies to more concrete security principles that can be effectively implemented and tested for a System of Interest (SoI). This work emphasizes the "design-for" purpose of the security principles, provides security requirements traceability, and points towards evidences of trustworthiness (e.g., design artifacts, analyses, test results).

II. SYSTEMS SECURITY ENGINEERING (SSE)

This section covers key SSE definitions and concepts to provide context for the reader seeking to further understand the specialty engineering discipline of SSE. Moreover, this background is essential for understanding the application of systems security design and architectural principles detailed in Sections III and IV.

A. SSE Definitions

While there is renewed interest in SSE within the United States Department of Defense (U.S. DoD) and broader industry, there is little discussion of formal definitions [2], [8]. In 1989, the U.S. DoD offered the first formal definition of SSE in Military Standard 1785 (now MIL-HDBK 1785) [9]:

> An element of system engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities. It uses mathematical, physical, and related scientific disciplines, and the principles and methods of engineering design and analysis to specify, predict, and evaluate the vulnerability of the system to security threats.

While this definition describes today's technical security concerns well, MIL-HDBK 1785 also includes a SSE Management definition, which reflects elements of the U.S. DoD's modern view of the technical and technical management aspects of systems engineering [9]:

An element of program management that ensures system security tasks are completed. These tasks include developing security requirements and objectives; planning, organizing, identifying, and controlling the efforts that help achieve maximum security and survivability of the system during its life cycle; and interfacing with other program elements to make sure security functions are effectively integrated into the total system engineering effort.

B. *Foundational SSE Concepts*

1) *SSE Informed System Development*

Figure 1 illustrates an SSE approach where those charged with executing SSE responsibilities are required to communicate up to key Stakeholders, down to security specialty areas, and *across* various Subject Matter Expert (SME) areas. While domain-specific security specialists must be able to identify and analyze vulnerabilities, a systems-oriented view of security requires a holistic view of the SoI. Thus, those responsible for executing SSE responsibilities need to be astute enough to understand the technical aspects of the broad continuum of security, inclusive of its physical, machine, technical, "cyber", and operational aspects, while simultaneously wielding enough programmatic wherewithal to intelligibly advise on the planning, development, and fielding of complex systems. An integrated SSE approach ensures sound security methodologies, processes, and best practices are considered throughout the entire system life cycle to meet the Stakeholders' security needs and objectives.

2) *A Standardized Engineering Approach*

Standardized engineering approaches, such as ISO/IEC/IEEE 15288, facilitate shared understanding amongst multiple Stakeholders, Engineers, and various other specialty disciplines [10]. For example, ISO/IEC/IEEE 15288 defines six common life cycle stages (Concept, Development, Production, Utilization, Support, and Retirement), which promote the consistent development of unprecedented systems with defined entry/exit criteria, expected artifacts, and well-understood decision points [11]. Furthermore, this vetted standard has 30 defined engineering processes which are used to provide rigor throughout the six engineering life cycle stages. With respect to system security, the use of a standardized approach aims to reduce shortcomings in the allocation of responsibilities between Stakeholders, Systems Security Engineers, and domain-level security SMEs though the utilization of well-defined life cycle stages and associated processes [8], [12].

3) *Applicable to Multiple System Types*

SSE personnel must also be equipped to address system-level security considerations in a number of common application domains. Moreover, SSE is increasingly becoming a necessary undertaking across many system types [13]. Thus, an SSE approach which is not industry-specific nor focused exclusively on "cybersecurity" is required. This means a system agnostic SSE approach which is capable of leveraging established technical and non-technical processes to achieve cost-effective security solutions across various system types regardless of their intended purpose(s), application domain(s), technological implementation(s), end user(s), or operational environment(s).

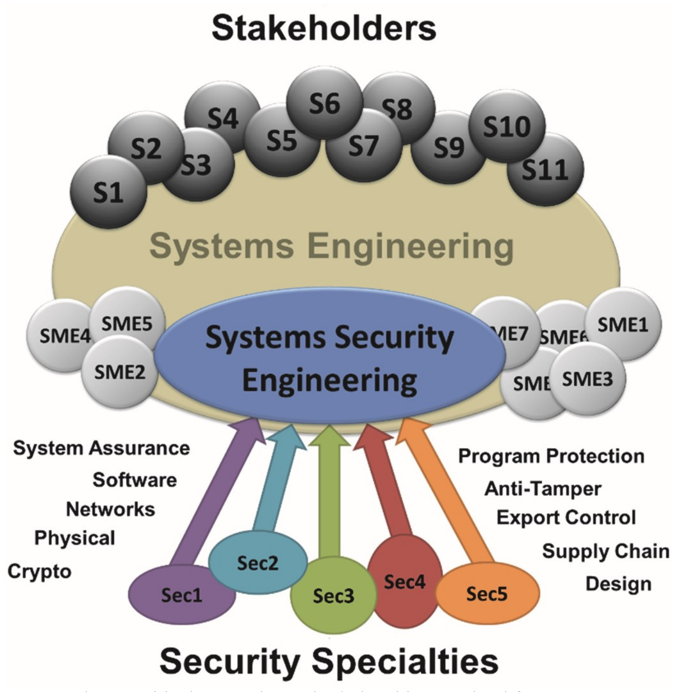

Fig. 1. Critical SSE roles and relationships. Derived from [14].

III. Secure System Development Strategies

Although there are many security best practices available in the literature, there is relatively little recent work which focuses on unifying security strategies [15]; thus, we would like to thank Paul Clark, Cynthia Irvine, and Thuy Nguyen for their research contributions which form the basis of the NIST SP 800-160 Appendix F. Please note that much of the Section III and IV discussion is largely derived from Appendix F with supplemental commentary to emphasize the system-level applicability of the security strategies and principles to a multitude of system types and personnel. Appendix F describes three overarching systems security development strategies [1]:

1. Access Mediation (The Reference Monitor Concept)
2. Defense in Depth
3. Isolation – Physical and Logical

These NIST-provided strategies offer an excellent starting point for framing the discussion of understanding and applying systems security design principles.

A. *Access Mediation (The Reference Monitor Concept)*

The access mediation strategy provides a conceptual model of the necessary access controls (or rules) that must be achieved to enforce security policies. This strategy is critically important because it constitutes the core fundamental concept in security design [16]. Thus, access mediation builds the foundation to successfully execute SSE activities and tasks, including detailed security analyses of the SoI's design, architecture, and implementation (for further details please see [17]).

Ideally, realizations of the access mediation concept possess three properties [18]: (1) it is tamper-proof; (2) it is always invoked; and (3) it can be subjected to analysis and testing to assure correctness (i.e., the reference validation mechanism). This means that any mechanism (physical or logical) claiming to perform access mediation only does what it is supposed to do and can never be bypassed, coerced, manipulated, or fooled. These properties also assume that the security policy can be correctly defined by humans and accurately transformed into a representation understood by the mechanism.

While fully acknowledging that these properties are not achievable in real-world systems, the intent of access mediation is primarily to help developers think critically about security mechanisms; and thus, avoid ad hoc security approaches. Ultimately, the technical constrains and practical limitations of realized security solutions translate into risks for the SoI. Since it is critical to manage these risks, those charged with SSE roles and responsibilities need to be able identify, understand, and mitigate system-level issues in a systematic way.

B. *Defense in Depth*

Perhaps the most plainly understood systems security strategy, defense in depth describes security approaches (i.e., conceptual and physical architectures) which create a series of barriers to prevent, delay, or deter an attack by an adversary. Typically, defense is depth is achieved through the application of multiple security mechanisms. While the application of multiple security components may increase system assurance, there is no formalized theoretical basis to assume that defense in depth alone achieves a level of trustworthiness greater than that of the individual security components. Additionally, it is important to note that implementing a defense in depth strategy is not a substitute for or equivalent to a choosing a sound security architecture or system design that leverages a balanced application of security concepts and design principles [19].

C. *Isolation*

The isolation security strategy pertains to the creation of separated processing environments; they can be logical, physical, or a combination thereof. Logical isolation requires the use of underlying trustworthy mechanisms to minimize resource sharing. For example, domain separation is in commonly used in many workplaces separating user accounts from the underlying operating system. As another example, isolated computer environments can be easily created through virtualization. Despite increases in the use of virtualized environments, research continues to demonstrate that isolation for processing environments can be extremely difficult to achieve [20].

More concretely, physical isolation typically involves separation of components, systems, and networks by hosting them on discrete hardware components. Under the broader systems security purview, it is also important to note that isolation may include the use of specialized facilities and/or operational procedures to control personnel actions and access. Thus, in many operational environments, isolation objectives are achieved through a combination of logical and physical mechanisms. For example, in one of the most impressive cyber-physical attacks to date, Stuxnet, both physical and logical isolation boundaries were violated [21]. Thus, it is critically important for systems security engineers to be cognizant of co-dependencies between logical and physical mechanisms.

IV. Strategy to Principles Mapping

Based on decades of work, a number of security best practices, principles, and patterns have been proposed. For example, the National Security Agency (NSA) specifies nine security "first principles" in their educational criteria [22]. As another example, dozens of security patterns are captured in [23]. In a third example, the U.S. DoD's System Survivability Key Performance Parameter suggests three pillars and ten attributes to achieve cybersecurity and survivability [24]. While none of these approaches are inherently deficient, the NIST SP 800-160 uniquely captures the essence of these works in 18 well-defined systems security principles shown in Table I.

A. *Mapping Introduction*

Before addressing the security principles in detail, it is helpful to first consider our intentions – to provide system architects, designers and developers concrete engineering principles that can be designed-for, built-in, and tested to meet stakeholders' security needs and objectives. Shown in Figure 2, our mapping illustrates the various relationships between the security strategies and principles [25]. This mapping allows users to more easily understand the complexities associated with implementing the security principles [26]. For example, by reading down each column, the developer can ascertain which design principles inform the desired security strategy or principle. Reading across each row, the developer can see which security principles contribute to related security strategies and principles. Most importantly, examining each row, Figure 2 brings insight into the tradeoffs associated with the application of each security principle – there are inherent conflicts and contradictions that must be considered (knowingly or unknowingly) when applying the security principles.

These mappings enable reasoning about, and justification of, security design decisions which point towards evidences of sound SSE strategy selection and implementation. For example, stakeholders, auditors, and security specialists can consider which security principles should be chosen, their implementation, and security assessments. Lastly, these mappings promote requirements traceability, support decisions of trustworthiness, and provide justification of limited resources. In the following subsections, each of the 18 design principles is systematically discussed one by one.

1) *Clear Abstractions*

This principle promotes readily understandable system-level abstractions such as subsystem elements, orderly data objects, and cohesive logical groupings which provide the engineer insight into the SoI's design. This in turn helps to achieve shared understanding amongst multiple stakeholders and facilitate more effective design reviews. Well defined abstractions also assist security engineers and decision makers in conducting security analysis and testing activities. Please note that while concepts such as "clarity" are inherently subjective, modeling languages such as SysML and UML provide rapidly maturing and standardized approaches for defining and communicating abstractions [27].

Clear abstractions support several security strategies and principles, but are particularly important for access mediation and isolation strategies. This due to their importance in defining relationships and behaviors between various subsystem elements, functions, users, dependencies, and data exchanges (i.e., all the SoI's important objects and their associations). For example, information dependencies must be detailed before access control rules can be appropriately defined. On a related note, clear abstractions also imply that such dependencies can be fully known across the SoI's users, subsystems, components, supporting/enabling systems, and various forms of data.

2) Least Common Mechanism

Longstanding design guidance such as "high cohesion" & "low coupling" are implicit in this design principle as like functionality is singularly consolidated, which reduces the security related development and analysis effort [28]. In general, a single instantiation of a mechanism (security or otherwise) allows for more efficient use of limited resources throughout the system lifecycle. Ideally, this principle also contributes to the development of more effective security mechanisms, since the design and engineering effort can be more focused.

Both access mediation and isolation strategies are implicit in this principle, while also reducing system complexity. Implementing a single mechanism can significantly reduce the complexity of access control rules, and thus, supports both reduced lifecycle costs as there are not multiple instantiations to document and modify.

3) Modularity and Layering

Modularity and layering are fundamental principles across both systems engineering and software disciplines [8], [28]; they serve to increase understandability by logically structuring and delineating dependencies between functional entities and data structures. The essential role of this design-focused principle is reflected in its direct support to several security strategies and principles to include access mediation, isolation, minimal complexity, hierarchical decomposition, and separation of applications into specific domains.

Commonly, this principle is used to facilitate the realization of security policy by restricting privileges of users, functions, and entities (i.e., access control). It is also important to recognize that resiliency is often achieved through modularity and layering which limits the damage inflicted by an attack or failure. For example, a secure system architecture minimizes dependencies and interactions such that when a component is compromised, it does not render other mission essential functionality inoperable. Lastly, it is worth stating that "layering" is not the same as "defense in depth" – the former is focused on the efficient design of a system, while the latter is focused on redundant means for protecting a system.

4) Ordered Dependencies (Partially)

Although not entirely necessary, the term "partially" is typically included in the principle title to imply that not all layers (and modules) can be strictly ordered. More simply, this principle suggests that higher layers should depend on lower layers and multi-layer circular dependencies should be avoided.

Logically structuring and minimizing dependencies contributes to isolation between layers, increases understandability of the design, reduces system complexity in the implementation, and facilitates test and analysis. A system

TABLE I
DESIGN PRINCIPLE DEFINITIONS DERIVED FROM NIST SP 800-160, APPENDIX F [5].

Principle Name	Definition (note, descriptions are slightly modified from NIST SP 800-160 to emphasize system-level applicability)
Clear Abstractions	A system should have simple, well-defined interfaces and functions to provide a consistent and intuitive view of the SoI's data, data elements, and how the data is utilized and managed.
Least Common Mechanism	If multiple components in a system require the same functionality (e.g., a necessary security feature), the desired functionality should be built into a single mechanism (physical or logical) which can be used by all components who require it.
Modularity and Layering	Modularity organizes and isolates functionality and related data flows into well-defined logical groupings (conceptual elements or "objects"), while layering orders and defines relationships between entities and their associated data flows.
Ordered Dependencies (Partially)*	Ordered dependencies refers to the logical arrangement of layers (and modules) such that linear (or hierarchical) functional calls, synchronization, and other dependencies are achieved, and circular dependencies are minimized.
Efficiently Mediated Access	Policy enforcement mechanisms (physical and logical) should utilize the least common mechanism available while satisfying stakeholder requirements within expressed constraints.
Minimized Sharing	No resources should be shared between system components (e.g., elements, processes, etc.) unless it is absolutely necessary to do so.
Reduced Complexity	The system design should be as simple and small as possible.
Secure Evolvability	A system should be developed to facilitate secure maintenance when changes to its functionality, architecture, structure, interfaces, interconnections, or its functionality configuration occur.
Trusted Components	A component must be trustworthy to at least a level commensurate with the security dependencies it supports.
Hierarchical Trust	Building upon the principle of trusted components, hierarchical trust provides the basis for trustworthiness reasoning when composing a system from a variety of components with differing trustworthiness.
Commensurate Protection*	The degree of protection provided to a component must be commensurate with its trustworthiness – as the trust placed in a component increases, the protection against unauthorized modification of the component should increase to the same degree.
Hierarchical Protection	A component need not be protected from more trustworthy components.
Minimize Trusted Components*	A system should not have extraneous trusted elements, components, data, or functions.
Least Privilege	Each system element (e.g., enabling systems, components, data elements, users, etc.) should be allocated sufficient privileges to accomplish its specified function, but no more.
Multi-Factor Permissions*	Requiring multiple authorizing entities or operators to provide consent before a highly critical operation or access to highly sensitive data, information, or resources is granted.
Self-Reliance*	Systems should minimize their reliance on other systems, elements, or components for their own trustworthiness.
Secure Composition*	The composition of various components that enforce the same security policy should result in a system that enforces that policy at least as well as the individual components do.
Trusted Communication	Each communication channel (i.e., an interface, link, or network) must be trustworthy to a level commensurate with the security dependencies it supports.
* denotes that the principle's name has been slightly modified to improve understandability and applicability for a broad developmental audience.	

Legend
"●" indicates a strong positive relationship
"o" indicates a weak positive relationship
"–" indicates a conflicting relationship
"X" indicates a relationship that could be either positive or negative

		Security Strategies			Structural Security Principles																	
		Access Control	Defense in Depth	Isolation	Clear Abstractions	Least Common Mechanism	Modularity and Layering	Partially Ordered Dependencies	Efficiently Mediated Access	Minimized Sharing	Reduced Complexity	Secure Evolvability	Trusted Components	Hierarchical Trust	Commensurate Protection	Hierarchical Protection	Minimized Security Elements	Least Privilege	Proportional Permissions	Self-Reliant Trustworthiness	Secure Distributed Composition	Trusted Communication Channels
Security Strategies	Access Control			●	Intentionally left blank																	
	Defense in Depth			o																		
	Isolation	●																				
Structural Security Principles	Clear Abstractions	●	o	●		●	o	o	●	●	●	●	o	o	o	o	●	●	o	o	o	o
	Least Common Mechanism	●		●			o	X	o	–	●	●		o			●	o				
	Modularity and Layering	●	●	●		o		●	●	o	●	●					o				o	
	Partially Ordered Dependencies			o			●				●			●		●						
	Efficiently Mediated Access	●		●		●	o			●	●											
	Minimized Sharing	●	o	●		–	●	●			●	o					o					
	Reduced Complexity	●	–	X	o	o	●	●	●	●		●	●	o	●	●	●	●	o	●	●	●
	Secure Evolvability	–	o	o	o	●	●	●	●	●	o		o	o	o	o	●	o	o	o	o	o
	Trusted Components		●				o	o						●	●	o					●	
	Hierarchical Trust	●	o					●							●	●						
	Commensurate Protection	●	●	o										●		o	–	o	o	●	o	●
	Hierarchical Protection	●	o	o			o													o		
	Minimized Security Elements	o	–	o		●	o		●	●	●	o									o	o
	Least Privilege	●		●					o	●	o	o	●				●		o	●		
	Proportional Permissions	●	o	–							●	o										
	Self-Reliant Trustworthiness			●	o	●	●		●	●												
	Secure Composition	o	●	–	o	–	o	●	o	●	–	X	●	●	●	●	●	●	●	●		
	Trusted Communication	●	o	●	o		o		●	o		o	●	●							●	

Fig. 2. Mapping of System Security Strategies to Design Principles.

with partially ordered dependencies is also less likely to negatively impact associated functions and elements, thus contributing to survivability. Additionally, partially ordered dependencies helps to preserve trustworthiness by avoiding linkages between components with lower and higher trust levels.

5) Efficiently Mediating Access

Efficiently and effectively mediating access is typically achieved through an operating system configured to enforce a policy on the use of system resources. For example, preventing users from accessing privileged or non-essential system functionality serves to protect the user(s) and SoI. Moreover, due to the principles of least common mechanism much of the desired protection capability is resident in a single mechanism which can contribute to performance bottlenecks. Thus, the design engineer must carefully consider the means for mediating access (e.g., technological solutions, process constraints, personnel restrictions, etc.) as to not negatively impact system performance or the desired protection capability.

This principle is directly related to the realization of the access mediation and isolation strategies to include people, processes, and technologies. Additionally, it is related to a number of other security principles such as minimizing sharing, trust, and hierarchical structures. It is also worth noting that while verification and validation should be accounted for in the development of sound security principles, often these requirements are not sufficiently addressed; thus, independently addressing the application of this principle is important because analytically evidences are used to substantiate claims of trustworthiness which support the achievement of security objectives, as well as, determinations of risk(s).

6) Minimized Sharing

In contrast to modern trends of sharing system resources and access to information, this principle stresses limited sharing of system resources (e.g., hardware, software, data, people, etc.). In particular, the sharing of data outside the SoI merits close scrutiny to avoid unauthorized access, disclosure, use, tampering, or modification. Internal to the system, developers must be mindful as to not hinder mission essential functions during diligent application.

This principle directly supports the access mediation and isolation strategies by reducing the number of interactions between users and system elements. Curtailing the sharing of resources effectively creates boundaries within the SoI to protect critical functions, simplify design and implementation (minimizing the attack surface), and facilitates defense in depth. While limiting shared resources may be advantageous for security reasons, this principle may be viewed as being in conflict with other security strategies and principles such as least common mechanism, so the benefits of each must be fully considered.

7) Reduced Complexity

There are several potential benefits to simpler systems including: increased understandability, ease of analysis, less prone to errors, fewer vulnerabilities, and lower costs. Moreover, these benefits can be realized across the entire development lifecycle (i.e., from concept to implementation, operation, sustainment, and retirement) which enables the desired security policy to be more effectively achieved.

Reduced complexity is particularly important for assessing access mediation mechanisms to demonstrate that the desired protection capability is achieved. Clarity of design enables the identification of potential vulnerabilities during testing (i.e., verification and validation) and their associated mitigations. Because of the additional insight gained through simplicity, this principle strongly contributes to nearly every design principle. Additionally, simpler designs help facilitate isolation through clarity of design and minimizing unnecessary interdependencies which can also contribute to system survivability (note, taken too far simplicity can serve to eliminate isolation mechanisms).

8) Secure Evolvabilty

This principle addresses both planned and unplanned updates, modifications, reconfigurability, and agility. Although primarily executed during operations and maintenance phases of the lifecycle, the system must be carefully designed and engineered to facilitate secure modifications. For example, if change is "planned-in" from the conceptual phase, secure evolvability can be "designed-in" to produce a more robust system that is built to be upgraded in a secure manner rather than ad hoc patching and after-the-fact solutions which attempt to adapt to the changing threat and operational environment [29].

Acknowledging that complex systems often have long lifecycles and face a dynamic set of constantly evolving threats, this principle is key to improving system security and survivability. More specifically, it is critical for facilitating secure software and hardware upgrades, modifications, and patches during operations and sustainment. Thus, this principle supports nearly all security strategies and principles throughout the SoI's lifecycle. Focusing on secure evolvability early in the lifecycle also has the potential to drive down engineering costs and facilitate less complex mitigations to future threats; however, it can negatively impact access mediations because of poor design and implementation choices.

9) Trusted Components (and Functionality)

This principle highlights that the trust chain is only as strong as its weakest link; thus, each component's trustworthiness must be commensurate with the broader SoI's desired trustworthiness for a given security functionality. While "trust" is often applied to low-level physical components, this security principle is equally applicable to subsystems, mission critical functionality, operators and support personnel, logistical activities, and communication channels.

This principle is foundational for the development of assured systems and operations as it facilitates reasoning about the evidences for decisions of trustworthiness, and more specifically it highlights where the SoI's trust chain is being limited by less trustworthy link(s). More concretely, trusted components (i.e., their functionality) enable the construction of trustworthy secure systems such that trustworthiness is not inadvertently diminished or misplaced as described in the hierarchical trust principle and supported by other structural-oriented principles.

10) Hierarchical Trust

The principle of hierarchical trust builds on the principle of trusted components and stresses the need to look vertically along trust dependency chains to ensure lower trust components (and functions) do not diminish the system's overall security posture. More generally, hierarchical trust can be interpreted as the "architecting" of a trustworthy system from a variety of components with differing trust levels. For example, if a more trustworthy component depends upon a less trustworthy component, this would in effect, put the components in the same "less trustworthy" equivalence class per the principle of trusted components. Formally, the system forms a "partial ordering" if it preserves the principle of trusted components.

Hierarchical trust is essential to achieving (and reasoning about) the trustworthy security of complex systems composed of various components (and dependencies) at differing levels of trustworthiness. In particular, the trust principles (#9 and #10), along with partial ordering, provide evidences for reasoning about and justifying trustworthiness decisions. Within the context of NIST SP 800-160, the hierarchical trust principle supports the achievement of trustworthiness in the security strategies and depends upon the correct implementation of several design principles.

11) Commensurate Protection

Formally named "inverse modification threshold", this principle suggests that the degree of protection provided to a component (or security function) should be commensurate with its trustworthiness. Thus, the higher a component's/function's value to the trust chain, the more protection it should warrant.

This principle expressly builds on the principles of trusted components and hierarchical trust, and is supported by several other security principles. This principle specifically contributes to access mediation and defense in depth strategies, and indirectly supports many principles as well. By deliberately focusing on the SoI's most critical components and functions, this principle ensures that adequate trustworthiness is designed into the system with supporting evidences.

12) Hierarchical Protection

While relatively straightforward, this principle is important for properly scoping and bounding the security engineering effort. Approaches such as the U.S. DoD's criticality analysis can be used to help focus limited security resources [12]; for example, the most critical components (i.e., those which execute mission essential functions) in a combat weapon system must be protected from other less trustworthy components. It is also worth noting that operators and other personnel should also be considered in the application of this principle.

This principle supports all three strategies by protecting the SoI from untrusted components, functions, data, and users with lower trustworthiness and reserving higher level privileges for more trusted entities. Regarding security design decisions, this principle also guides the application of architectural and component level principles to ensure security resources are not wasted on protections against higher trust level components.

13) *Minimize Trusted Components*

This principle suggests that the SoI should contain as few trustworthy components (or systems elements) as possible. Somewhat counter-intuitive to security practitioners, when implemented properly this principle discourages extra security components, features, and technologies (which are also likely to introduce additional vulnerabilities).

Minimizing the number of trusted components simplifies the testing associated with access mediation and other protection principles such as hierarchical trust. Much like hierarchical protection this principle also helps to minimize costs and complexity of the desired protection capability. Arguably, the resulting simplicity also enhances survivability.

14) *Least Privilege*

While "least privilege" is typically understood as the granting and revoking of user privileges during a system's operation, it is an essential design principle for system security development in terms of the SoI's internal structure and organization, as well as, the providing capability for the assignment of user privileges. By allocating only the minimum privileges necessary to each component, when one is misused, damaged or compromised the impact to the system will be limited by the scope of the privileges granted. Additionally, it is important to note that this principle has widespread applicability for securing both internal and external interactions (e.g., system elements, supporting/enabling systems, data processing, data storage).

Thus, the principle of least privilege is pervasive and directly supports access mediation, often with the outcome of logical or physical isolation. This helps to minimize the impact of potential failures, corruption, misuse, and malicious activities. Least privilege also serves to reduce interdependencies, which simplifies component design, implementation, and analysis. Additionally, applying the principle of least privilege can improve survivability as an attacker's movements are limited when they are denied privilege escalation, a key tactic employed by advanced persistent threats.

15) *Proportional Permissions*

Somewhat akin to two-person authentication—a well-established security best practice in multiple domains such as financial [19] and flight safety [30]—the use of multiple individuals, organizations, or system entities to grant access decreases the likelihood of abuse and provides additional protection that no single accident, deception, or breach of trust is sufficient to enable an unrecoverable action. Note, although typically thought about as parallel authorization, proportional permissions can be serial in nature across two or more entities.

This principle directly contributes to the access mediation strategy and defense in depth, while also enhancing survivability by protecting mission critical information, components, and processes. However, it is also important to note that relying on multiple valid authentications can also negatively impact availability and/or survivability. Lastly, the predicate permissions principle adds complexity into the system design, so its usage needs to be considered carefully within the engineering trade space.

16) *Self-Reliance*

In addition to minimizing the SoI's reliance on other systems for its own trustworthiness, this principle should also be applied to subsystem elements, objects, and functions which require high levels of trustworthiness. Application of this principle minimizes the number of dependencies (e.g., supporting/enabling systems, data feeds, and personnel interactions). Perhaps, the importance of this principle is best illustrated with a counterexample where system developers often assume trustworthiness of input data, which in many cases is unmerited and ultimately degrades the SoI's trustworthiness.

Primarily, this principle serves to isolate the SoI which reduces its attack surface and eliminates the system's susceptibility to vulnerabilities inherent in external links/systems, especially those it cannot control. If the principle is applied within the system's architecture, it can also serve to isolate mission critical functions and components. Self-reliance can also help to reduce design and implementation complexity, increase testability and improve system survivability.

17) *Secure Composition*

Formally titled "Secure Distributed Composition", this principle mitigates undesirable emergent security behaviors resulting from interactions across the SoI's functions, components, and supporting/enabling systems. To ensure the desired system-wide policy enforcement is correct, the SoI must be thoroughly understood and analyzed; this is particularly important for distributed system architecture built from heterogeneous systems.

This principle is related to all of the security strategies and principles but is strongly dependent upon the hierarchical trust and hierarchical protection principles to ensure commensurate implementation of security policy across isolated systems (and their respective permission levels). Because this principle includes the composition of distributed features, components, and system elements, it necessarily touches nearly all design principles. Application of this principle also exacerbates the security principles which pertain to communication between components where the developer must work to identify and assess emergent behaviors and properties.

18) *Trusted Communication*

For modern systems of interest, the trusted communication security principle is critically important, especially when considering advanced cyber-physical systems built to survive highly contested cyberspace environments. In contrast to focusing on the trustworthiness of physical components, this principle ensures protection is more adequately considered at the SoI's most susceptible points – its communication links. This is because communication channels are generally available to adversaries for eavesdropping, reverse engineering, and tampering which can negatively affect data availability and integrity. Formally, this principle requires that each communication channel be trustworthy to a level commensurate with the security functions it supports.

This principle is similar and related to a number of other design principles and is often achieved through conventional protection mechanisms such as encryption, which also contributes to access mediation and isolation security strategies. In particular, this principle ensures weaknesses are not introduced via susceptible communication channels and ensuring communication channels have the same level of protection as the components they support.

V. Conclusions

This work examines the NIST SP 800-160 systems security strategies and design principles, and more specifically offers a mapping of conceptual strategies to concrete security principles that can be more effectively designed-for, built-in, and tested. This work is part of a series of works which aims to assist developers, owners, and operators in understanding and achieving a rigorous SSE approach. Future work includes studying the efficient application of these principles and their applicability to cyber resiliency, as well as identifying appropriate technical performance measurements and criteria.

Acknowledgements

We would like to thank Michael A. McEvilley for his time, review, and contribution of this work. This acknowledgement does not convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the authors.

References

[1] R. Ross, M. McEvilley and J. C. Oren, "Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems," NIST, 2016.

[2] K. Baldwin, J. Miller, P. Popick and J. Goodnight, "The United States Department of Defense Revitalization of System Security Engineering Through Program Protection," in *IEEE Systems Conference*, 2012.

[3] D. Snyder, J. D. Powers, E. Bodine-Baron, B. Fox, L. Kendrick and M. H. Powell, "Improving the Cybersecurity of U.S. Air Force Military Systems Throughout Their Life Cycles," RAND Corporation, 2015.

[4] National Defense Industrial Association, "A Path Towards Cyber Resilient and Secure Systems," NDIA Systems Engineering Division, 2016.

[5] L. O. Mailloux, M. A. McEvilley, S. Khou and J. M. Pecarina, "Putting the "systems" in security engineering: an examination of NIST special publication 800-160," *IEEE Security & Privacy,* vol. 14, no. 4, pp. 76-80, 2016.

[6] S. Khou, L. O. Mailloux and J. M. Pecarina, "System-agnostic security domains for understanding and prioritizing systems security engineering sfforts," *IEEE Access,* vol. 5, pp. 3465-3474, 2017.

[7] S. Khou, L. O. Mailloux, J. M. Pecarina and M. A. McEvilley, "A customizable framework for prioritizing systems security engineering processes, activities, and tasks," *IEEE Access,* vol 5. pp. 12878-12894, 2017.

[8] International Council on Systems Engineering, "INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 4," 2014.

[9] United States Department of Defense, "System security engineering program management requirements," 1989.

[10] ISO/IEC/IEEE, "Systems and software engineering — System life cycle processes, Third Edition," Geneva, Switzerland, 2015.

[11] ISO/IEC TS, "Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management, Second Edition," Geneva, Switzerland, 2016.

[12] Defense Acquisition University, "Defense Acquisition Guidebook," 05 April 2017. [Online]. Available: https://www.dau.mil. [Accessed 04 May 2017].

[13] S. Dietrich, "Cybersecurity and the Future," *IEEE Computer,* vol. 50, no. 04, p. 7, 2017.

[14] L. O. Mailloux, C. Garrison, R. Dove and R. C. Biondo, "Guidance for Working Group Maintenance of the Systems Engineering Body of Knowledge (SEBoK) with Systems Security Engineering Example," in *INCOSE International Symposium*, 2015.

[15] C. Irvine and T. D. Nguyen, "Educating the Systems Security Engineer's Apprentice," *IEEE Security & Privacy,* vol. 8, no. 4, pp. 58-61, July/August 2010.

[16] C. E. Irvine, D. F. Warren and P. C. Clark, "The NPS CISR graduate program in infosec: Six years of experience.," in *Naval Postgraduate School Monterey CA Dept of Computer Science*, 1997.

[17] M. A. Bishop, The art and science of computer security, Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[18] C. E. Irvine, "The Reference Monitor Concept as a Unifying Principle in Computer Security Education," Dept of Computer Sci., Naval Postgraduate School, Monterey CA, 1999.

[19] R. Anderson, Security Engineering, 2nd ed., Indianapolis, Indiana: Wiley Publishing, Inc, 2008.

[20] A. Van Cleeff, W. Pieters and R. Wieringa, "Security implications of virtualization: A literature study," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, 2009, August.

[21] R. Langner, "To kill a centrifuge: a technical analysis of what stuxnet's creators tried to achieve," 2013.

[22] National Security Agency, "National Centers of Academic Excellence," 16 November 2016. [Online]. Available: https://www.nsa.gov/resources/educators/centers-academic-excellence/cyber-operations/requirements.shtml. [Accessed 30 May 2017].

[23] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad, Security patterns: integrating security and systems engineering, John Wiley & Sons, 2013.

[24] Defense Acquisition University, "System survivability key performance parameter," 23 May 2017. [Online]. Available: https://dap.dau.mil. [Accessed 1 June 2017].

[25] D. V. Steward, "The design structure system: A method for managing the design of complex systems," *IEEE transactions on Engineering Management,* vol. 3, pp. 71-74, 1981.

[26] E. Crawley, B. Cameron and D. Selva, System architecture: Strategy and product development for complex systems, Prentice Hall Press, 2015.

[27] S. Friedenthal, A. Moore and R. Steiner, A practical guide to SysML: the systems modeling language, Morgan Kaufmann, 2014.

[28] C. Larman, Applying UML and Patterns, Third Edition ed., Pearson Education, Inc., 2005.

[29] S. R. Goerger, A. M. Madni and O. J. Eslinger, "Engineered resilient systems: A DoD perspective," in *Procedia Computer Science*, 2014.

[30] N. Leveson, "A new accident model for engineering safer systems," *Safety science,* vol. 42, no. 4, pp. 237-270, 2004.