IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 5, 2019, accepted July 12, 2019, date of publication July 24, 2019, date of current version August 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930718

Analysis of Systems Security Engineering Design Principles for the Development of Secure and Resilient Systems

PAUL M. BEACH[1], (Student Member, IEEE), LOGAN O. MAILLOUX[1], (Senior Member, IEEE), BRENT T. LANGHALS[1], AND ROBERT F. MILLS[2], (Senior Member, IEEE)

[1]Department of Systems Engineering and Management, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH 45385, USA
[2]Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH 45385, USA

Corresponding author: Logan O. Mailloux (logan.mailloux@us.af.mil)

This work was supported in part by the Air Force Research Laboratory Space Vehicles Directorate (AFRL/RV).

ABSTRACT The increasing prevalence of cyber-attacks highlights the need for improved systems security analysis and engineering in safety-critical and mission-essential systems. Moreover, the engineering challenge of developing secure and resilient systems that meet specified constraints of cost, schedule, and performance is progressively difficult given the trend toward increasing complexity, interrelated systems-of-systems. This paper analyzes the 18 design principles presented in the National Institute of Standards and Technology Special Publication (NIST SP) 800-160 Volume 1 and considers their applicability for the development of secure and resilient systems of interest. The purpose of this work is to better understand how these design principles can be consistently and effectively employed to meet stakeholder defined security and resiliency needs as part of a comprehensive systems security engineering approach. Specifically, this work uses the Design Structure Matrix (DSM) analysis to study the 18 design principles presented in NIST SP 800-160 Vol. 1, Appendix F, along with their intra- and inter-dependencies to develop complex cyber-physical systems that are secure, trustworthy, and resilient. The DSM analysis results increase understanding of the various relationships between the 18 design principles and identifies two clusters for secure systems design: Architecture and Trust. Lastly, this work provides a notional command and control system case study, along with a detailed listing of engineering considerations, to demonstrate how these principles and their groupings can be systematically applied as part of a comprehensive approach for developing cyber-physical systems which are designed to operate in hostile environments.

INDEX TERMS Design principles, systems security engineering, security engineering.

I. INTRODUCTION

Modern systems are increasingly complex compositions of system elements, subsystems, supporting & enabling systems, and extensive infrastructures that often result in a myriad of cyber dependencies, complicated interactions, and emergent behaviors. Moreover, because of their cyber dependencies (e.g., hardware, software, communications, etc.) these expansive Systems-of-Systems are inherently susceptible to a wide range of malicious and non-malicious events which can result in unexpected disruptions and unpredictable actions. Substantiating these realities, the United States Department of Defense (U.S. DoD) – arguably the world's largest acquirer of complex systems – has conceded that they cannot confidently assure their critical mission systems will operate as intended through a full-spectrum cyber-attack by well-resourced nation-state actors [1]. Thus, it is necessary to undertake deliberate systems-level engineering efforts to design, develop, and field secure and resilient systems capable of operating 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]. Moreover, inherent in discussions of ''securing'' complex cyber-physical systems is often the issue of safety as a key stakeholder need [4]. For example, the safety of

The associate editor coordinating the review of this manuscript and approving it for publication was Lin Wang.

humans is paramount in the development of aircraft and more recently autonomous vehicles.

To address this critical systems security problem, the National Institute of Standards and Technology (NIST), the 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: NIST Special Publication (SP) 800-160, *Systems Security Engineering* [5]. Subtitled ''Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems,'' NIST SP 800-160 is focused on institutionalizing engineering-driven actions to develop more defensible systems through a rigorous analysis approach in alignment with industry standards for systems development such as ISO/IEC/IEEE 15288:2015 [6]. In 2018, NIST SP 800-160 was designated as NIST SP 800-160 Volume 1 with the draft release of NIST SP 800-160 Volume 2 [7]; however, at the time of this writing Vol. 2 is still in draft form and does not align with standardized systems engineering approaches (i.e., Vol. 1 is based on the widely accepted ISO/IEC/IEEE 15288 systems engineering standard [8] while Vol. 2 is based on the MITRE Cyber Resiliency Engineering Framework [9]). As a unified and comprehensive SSE approach, in this work, we've chosen to focus on the effective application of the systems security design principles as presented in NIST SP 800-160 Vol. 1.

In this work we extend our previous work [10] by performing Design Structure Matrix (DSM) analysis of the security-oriented design principles presented in NIST SP 800-160 Vol. 1 and studying their mappings to systems security strategies. The contribution of this paper is threefold:

1) Mapping the NIST SP 800-160 Vol 1. SSE design principles to the proposed resiliency security strategy
2) Studying the design principle interactions and clustering the principles for ease of implementation
3) Providing a case study which describes how the security and resiliency design principles can be applied to a given system

Section II of this paper introduces the broader context for this work as a standardized SSE approach and identifies seminal SSE related works. Section III details the DSM analysis methodology used to study the application of the subject security strategies and design principles. Section IV of this paper discusses the mapping of the 18 NIST SP 800-160 Vol. 1 design principles across the four SSE strategies in the design matrix, while Section V details the results of the DSM analysis into two clusters of secure systems design: Architecture and Trust. Additionally, in Section VI a notional command and control System of Interest (SoI) is used to provide a concrete example of how these principles and their groupings can be used to field secure and resilient systems. Of note, we also offer commentary on ''engineering considerations'' for developing mission critical systems.

Lastly, this work emphasizes the ''design-for'' purpose of the Vol. 1 security principles, provides security requirements traceability, and points towards evidences of trustworthiness (e.g., design artifacts, analyses, test results, etc.) which can aid system developers, owners, and operators in satisfying their security and resiliency needs by mapping conceptual strategies to security principles that can be implemented and tested.

II. A STANDARDIZED SYSTEMS SECURITY APPROACH

This paper is part of an ongoing research activity to understand and promote the application of a standardized approach for developing secure systems. More specifically, the ''systems security engineering'' approach adopted in this work is considered a specialty area of systems engineering and consistent with ISO/IEC/IEEE 15288, NIST SP 800-160 Vol. 1, and the International Counsel on Systems Engineering (INCOSE) handbook [5]. For related publications which seek to promote a systems-oriented view of SSE please see [11]–[14], and [15]; excellent comprehensive works are available here: [16], [17]. Note, while few formally adopt the name ''systems security engineering'' for their work the discipline of SSE continues to rapidly mature, especially with the ever increasing, world-wide interest in fielding secure and resilient cyber-physical systems such as autonomous vehicles.

A. DEFINING SYSTEMS SECURITY ENGINEERING

While the NIST SP 800-160 Vol. 1 does not formally provide a definition for SSE, two helpful definitions are offered in Military Handbook 1785 [18]:

> System Security Engineering: 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.
>
> System Security Engineering Management: 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.

With these two definitions in mind, the goal of SSE is to minimize system vulnerabilities to known and presumed security threats across the SoI's lifetime.

B. THE NIST SP 800-160 VOL. 1 DEVELOPMENT STRATEGIES AND DESIGN PRINCIPLES

Throughout the past several decades a number of security best practices, principles, and patterns have been proposed for system development. For example, the NSA specifies nine security ''first principles'' in their educational criteria [19].

In another example, dozens of security patterns are captured in [20]. 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 [21].

While none of these approaches are inherently deficient, the NIST SP 800-160 Vol. 1 uniquely captures the essence of these works and has been thoroughly refined in multiple rounds of public review [5]. Moreover, these strategies and principles are part of a compressive and standardized engineering approach. Thus, Vol. 1's strategies and principles are used to frame our analysis approach. The Vol. 1 systems security strategies design principles are described thoroughly in Section IV. For a detailed history of system-level security principles please see [22].

While the authors often reference works from the United States Government and Military, the topics discussed in this paper are broadly applicable to any organization seeking to develop secure cyber-physical systems of interest.

III. METHODOLOGY FOR MAPPING SYSTEMS SECURITY STRATEGIES TO DESIGN PRINCIPLES

As is consistent with our previous work and Vol. 1, we aim to maintain a unified, holistic, and standardized systems-based approach for systems security and resiliency. This ''top-down'' approach is in contrast to a ''bottom-up'' compliance-based cybersecurity perspective [6]. For example, while a highly skilled cybersecurity professional might know how to best secure a piece of hardware, the systems security engineer should be asking ''Why do we need to secure that piece of hardware?'' – there is a very important difference which can be subtle at times. The subtle difference between the two perspectives is important especially as modern systems become increasingly autonomous with high levels of trustworthiness and safety. In a proper SSE approach, the goal is to first understand the security and resiliency requirements for the CPS of interest and then design a suitable system. Unfortunately, the necessity of defining the security problem within the SoI's operational context can be missed [23].

A. STANDARDIZED APPROACH FOR SYSTEMS SECURITY AND RESILIENCY

Shown in Figure 1, a simplified system development model (known as the ''Vee model'') is useful for illustrating the basic premise of SSE – an initial investment in ''engineering-out'' vulnerabilities and ''designing-in'' security and resiliency countermeasures is a long-term cost saving measure [18], [24]. The Vee model depicted in Fig. 1 is overlaid with four critical SSE stages: Determining what to secure (blue), Designing a secure system (grey), Demonstrating the system is secure (green), and Analysis to support system security decisions (yellow). Most importantly, this approach is consistent with systems engineering best practices where security and resiliency stakeholder needs are identified early in the development life cycle (i.e., the blue processes) and their solution trade space is explored at reduced cost (i.e., the grey processes) [24]. Shown in green, testing and analytical evidences are used to support claims of trustworthiness, security, and resiliency.

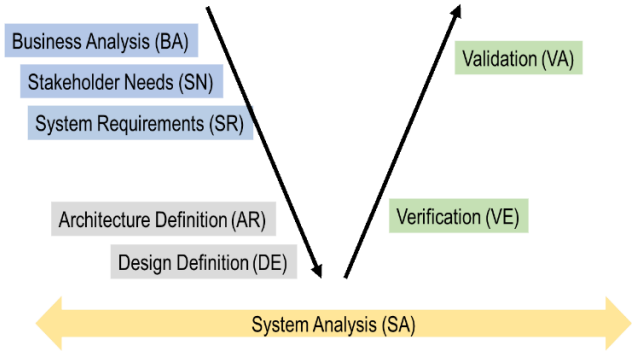

FIGURE 1. A simplified depiction of a standardized systems security engineering approach based on the systems engineering processes of ISO/IEC/IEEE 15288 [8].

Across the entire developmental life cycle, the systems analysis process is used to eliminate ad hoc approaches, utilize scientific reasoning, and infuse engineering rigor to systematically identify and reduce vulnerabilities (e.g., mathematical analysis, model and simulation, defined approaches, etc.). For a recent detailed example, please see [25]. While our previous works focused on understanding the blue SSE processes [5], [26]–[28], this work focuses on understanding the grey SSE processes through the analysis and application of the NIST SP 800-160 Vol. 1 principles.

B. INTRODUCTION TO DESIGN STRUCTURE MATRICIES

Shown in Figure 2, our strategy-to-principle mapping illustrates the relationships between the various security and resiliency strategies, and their associated principles. This mapping allows users to more easily understand the interdependencies associated with implementing these principles. Moreover, detailing these positive and negative relationships facilitates reasoning about important security design trade-offs. This mapping is also useful for producing evidences of sound engineering practices [29]. For example, stakeholders, auditors, and security specialists can more easily consider which security principles should be implemented given a security or resiliency strategy. Thus, these mappings can be used to support requirements traceability, decisions of trustworthiness, and justification of limited resources. Note, detailed discussion of the mapping is provided in Section IV.

More formally, Fig. 2 is a type of adjacency matrix known as a ''Design Structure Matrix'' or simply a DSM [30]. The intention of the DSM is to provide the reader with a visual depiction of the interactions between matrix elements (i.e., the strategies and principles list across the rows and columns). The DSM also takes advantage of directionality which allows additional information to be encoded in the matrix. For example, a developer can read down a particular column to ascertain which design principles inform or support the desired security strategy or principle (i.e., each marked row). Likewise, reading across each row, the developer can see how

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

Structural Security Principles	Security Strategies				Structural Security Principles																	
	Access Control	Defense in Depth	Isolation	System Resiliency	1. Clear Abstractions	2. Least Common Mechanism	3. Modularity and Layering	4. Ordered Dependencies (Partially)	5. Efficiently Mediated Access	6. Minimized Sharing	7. Reduced Complexity	8. Secure Evolvability	12. Hierarchical Protection	13. Minimize Trusted Components	16. Self-Reliance	9. Trusted Components	10. Hierarchical Trust	11. Commensurate Protection	14. Least Privilege	15. Multi-Factor Permissions	17. Secure Composition	18. Trusted Communication
1. Clear Abstractions	●	o	●			●	o	o	●	●	●	●	o	●	o	o	o	o	●	o	o	o
2. Least Common Mechanism	●		●	X			o	X	o	–	●	●					o		o			
3. Modularity and Layering	●	●	●	●		o		●	●	o	●	●		o							o	
4. Ordered Dependencies (Partially)			o				●				●						●					
5. Efficiently Mediated Access	●		●			●	o			●	●											
6. Minimized Sharing	●	o	●	o		–	●	●			●	o		o								
7. Reduced Complexity	●	–	X	X	o	o	●	●	●	●		●	●	●	●	●	o	●	●	o	●	●
8. Secure Evolvability	–	o	o	●	o	●	●	●	●	●	o		o	●	o	o	o	o	o	o	o	o
12. Hierarchical Protection	●	o	o	●			o								o							
13. Minimize Trusted Components	o	–	o	●		●	o		●	●	●	o									o	o
16. Self-Reliance			●	X	o	●	●		●	●	●											
9. Trusted Components		●		o				o	o				o				●	●			●	
10. Hierarchical Trust	●	o		o					●				●					●				
11. Commensurate Protection	●	●	o	●									o	–	●		●		o	o	o	●
14. Least Privilege	●		●	●						●				●	●	●						
15. Multi-Factor Permissions	●	o	–	X							–	o						●				
17. Secure Composition	o	●	–		o	–	o	●	o	●	–	X	●	●	●	●	●	●	●	●		
18. Trusted Communication	●	o	●	o	o		o	●		o		o				●	●				●	

FIGURE 2. Design Structure Matrix (DSM) provides a visual depiction of the positive and negative relationships between the various NIST SP 800-160 Vol. 1 systems security strategies and principles. The DSM takes advantage of directionality where a developer can read down a column to ascertain which design principles inform or support a particular security strategy or principle (i.e., each marked row). Likewise, reading across each row, the developer can see how each security principle contributes to other security strategies and principles. These details are particularly helpful for understanding tradeoffs between the security design principles. Lastly, DSM analysis results in two clusters with architectural principles shown in red and trust principles shown in blue.

each security principle contributes to other security strategies and principles. Examining the DSM brings insight into the tradeoffs associated with the application of each security principle as there are inherent conflicts and contradictions that must be considered when applying each security principle. Most importantly, the DSM enables a stochastic analysis of these relationships to provide an optimized clustering of the most interrelated elements.

C. DESIGN STRUCTURE MATRIX ANALYSIS

A DSM provides a means of modeling complex interactions in a set of elements and utilizes a systematic approach to analyze, order, and organize that information [30]. Thus, a DSM is a useful mechanism for studying the inter-relationships between the security design principles and strategies of Vol. 1.

Our DSM analysis builds upon the software developed by Thebeau and leverages his algorithm to perform the clustering of elements [31]. As shown in Fig. 2, the security principles are highly interrelated and suggest a densely populated DSM; thus, to provide the most value to the practitioner, only the strong positive relationships are fully examined (i.e., those with a solid circle "●"). Analyzing only strong relationships enables the optimization algorithm to categorize the security principles into useful groupings. Additionally, the relationships between the security strategies and the design principles are not used as inputs to the clustering algorithm because their inclusion within a single homogenous DSM would not be appropriate; thus, the four security and resiliency strategies are merely included to understand the applicability of the principles, and their clustering, as they pertain to achieving stakeholder needs.

The DSM optimization uses a stochastic bidding algorithm shown in Eq. (1) to determine which cluster a given element should be assigned to. This process is based on factors such as the size of the overall cluster and the number of interactions within a cluster where cost penalties are assigned to larger clusters to avoid returning a single cluster and promote the creation of highly inter-related clusters. To prevent localized optimum solutions, Thebeau's algorithm also incorporates simulated annealing which periodically accepts the second highest bid rather than the highest bid to increase the probability of finding a global optimum solution. The bid for a cluster j is given by:

$$ClusterBid_j = \frac{inout^{\text{powdep}}}{ClusterSize_j^{\text{powbid}}} \quad (1)$$

where: *inout* is the sum of DSM interactions of the chosen element with each of the elements in cluster j; powdep = 4 is a user-specified exponential to emphasize interactions; *ClusterSize* is the number of elements currently in cluster j; and *powbid* = 1 is a user-defined exponential to penalize the size of the cluster. Additionally, a *stable_limit* parameter in the control logic of the algorithm, which specifies how long the algorithm runs before reporting results, was adjusted. Specifically, the process was modified to perform longer and broader searches in order to return higher overall *likeliness* values which is largely due to the fact that the availability of computing capacity has dramatically increased since the original algorithm was coded in 2000. While the clustering algorithm permits clusters as small as a single unit or as large as the entire matrix, the *likeliness* value is introduced to measure the consistency of clustering results across multiple runs and increases the confidence in the repeatability of the outcomes. Our algorithm is available on the IEEE Access website as part of this publication. It is also worth noting that Borjesson and Hölttä-Otto improved the speed of the algorithm; however, given the relatively small size of this problem matrix, the original algorithm was more than sufficient when executed on modern hardware [32].

D. ADDITIONAL DEFINITIONS

Before starting a detailed discussion of the security principle analysis results, it is helpful to first introduce a few essential SSE terms from Vol. 1. Note, these definitions are not presented as such in the Vol. 1 but are constructed as ''working definitions'' to assist the reader in understanding the DSM analysis, and particularly, detailed discussion of protection and trust principles below.

> Protection: A capability (or statement of the stakeholder's security need/requirement) with the objective of controlling the events, conditions, and consequences that contribute to unacceptable losses.
>
> Trust: The degree to which the security behavior of a component is demonstrably compliant with its stated functionality.

Given the frequency of the concept in this discussion we also include a definition of trustworthiness to aid the reader [33]:

> Trustworthiness: Trustworthiness implies simply that something is worthy of being trusted to satisfy its expected requirements.

For the purposes of this study, we also adopt a definition of resilience from [1] to help frame the discussion and form a basis for analyzing the security principles in the broader context of SSE:

> Resilience: Resilience is the ability to continue or return to normal operations in the event of some disruption (natural or man-made, inadvertent or deliberate).

This concise definition is specific enough to provide a grounding for the following analysis yet broad enough to adequately cover a myriad of expected and unexpected events (e.g., malicious action, unintended misuse, system failures, etc.). For a more formal discussion of ''resiliency'' used in the study of complex networks please see [34].

IV. SECURITY PRINCIPLES MAPPING DISCUSSION

In this section, we discuss the 18 NIST SP 800-160 Vol. 1 security principles and their mappings as presented in Fig. 2. Specifically, we expand upon our previous work [10] to provide detailed explanations of each principle, rationale for mappings between the principles, introduce a fourth systems security strategy (Resiliency), and provide justification for the principle to strategy mappings. Ultimately, our goal with this work is to facilitate widespread adoption of these system-level security strategies and their associated principles such that defensible and resilient SoIs can be more easily designed, built, and tested to meet stakeholders' security needs regardless of the developer background or SoI's application domain. To facilitate this knowledge transfer, our DSM software and mappings are available on the IEEE Access website for the reader to study, modify, and improve upon.

A. SECURITY STRATEGIES

In this section, we formally introduce the systems security strategies studied in this work (1. Access Control, 2. Defense in Depth, 3. Isolation, and 4. System Resiliency). These four strategies have been modified and expanded upon those presented in NIST SP 800-160 Vol. 1, Appendix F [6] to improve understandability and broaden their applicability.

Shown in the green portion of Fig. 2, each of the four SSE strategies is supported by numerous design principles. The systems security strategies are presented in a one-way relationship within the DSM because the developer will primarily need to know which principles contribute to the desired security strategy (i.e., each upper-level strategy is supported by one or more principles). For example, if the stakeholders are developing a SoI that processes highly classified information, an isolation-focused security strategy may be desired. With the DSM of Fig. 2, the developer can easily see which security principles directly contribute (or in some cases take away) from the intended isolation-focused outcome.

1) ACCESS CONTROL

Access control (formally known as ''the Reference Monitor Concept'') provides a conceptual model of the necessary access controls (i.e., requirements) that must be achieved to enforce security policies. Ideally, realizations of the access mediation concept possess three properties: (1) tamper-proof; (2) always invoked; and (3) can be subjected to analysis and testing to assure correctness. This means that any mechanism claiming to perform access mediation only does what it is supposed to do and can never be bypassed, coerced, manipulated, or deceived.

2) DEFENSE IN DEPTH

Defense in depth describes security approaches which create a series of barriers to prevent, delay, or deter an attack by an adversary. Typically, defense in depth is achieved through the application of multiple security mechanisms (i.e., conceptual and physical obstructions). It is important to note that implementing a defense in depth strategy is not a substitute or equivalent to choosing a sound security architecture or system design that leverages a balanced application of security concepts and design principles [17].

3) ISOLATION (PHYSICAL AND LOGICAL)

Isolation pertains to the creation of separated processing environments; they can be logical, physical, or a combination thereof. While physical isolation is somewhat more straightforward (i.e., physically distinct components and systems), the sharing of data often necessitates logical isolation mechanisms which requires the use of underlying trustworthy mechanisms to minimize resource sharing.

4) RESILIENCY

In addition to the three security strategies of Vol. 1, we also considered emerging stakeholder needs such as ''cyber resiliency'' and ''system survivability'' which have generated congressional mandates, service-level policy changes, and significant organizational changes within the U.S. DoD [35]. Introduced in 2005, the System Survivability Key Performance Parameter (SS KPP) ensures the SoI can maintain critical functionality while under attack [21]. Formally, the SS KPP includes three pillars (Prevent, Mitigate, Recover) intended to reduce system susceptibility to adversaries, limit damage from exploited vulnerabilities, and increase system resiliency in order to execute the SoI's mission. Likewise, the U.S. DoD and similar industries are increasingly focusing on resiliency – the ability of a system to perform essential functions in spite of hostile actions [36].

Thus, we consider ''system resiliency'' or ''resiliency'' as an important system-level security strategy in addition to Vol. 1's three strategies. While the topic of ''resiliency'' is being studied in multiple domains from complex networks to biological [37], we thought it pertinent to include resiliency for advanced cyber-physical systems in our analysis as it is timely and germane to developing more secure and defensible systems. For additional background on the topic of systems-based resiliency, please see: [38]–[40]. Of note, during the course of our research, the draft version of NIST SP 800-160 Vol. 2, Cyber Resiliency, was released [7]. Vol. 2 is largely focused on countering the cyber-specific ''Advanced Persistent Threat'' as a manifestation of the MITRE Cyber Resiliency Engineering Framework (CREF) [38]. While Vol. 2, and the CREF, are very helpful in many regards, in this work we strive to maintain a standardized approach consistent with industry standards (i.e., ISO/IEC/IEEE 15288, INCOSE handbook, and specifically the security design principles of Vol. 1). More generally, we are interested in engineering resilient systems which are able to operate in and through situations where the SoI is degraded, whether due to hostile actors (insiders or outsiders) or internal failures (hardware or software).

B. DESIGN PRINCIPLES

In this section, we formally introduce the 18 SSE principles from NIST SP 800-160 Vol. 1, Appendix F [6]. In particular, we've modified and extended these principle descriptions to make them less ''computer science centric'' and broaden their applicability to any engineering discipline at the ''systems level'' of thinking. More pragmatically, we've attempted to make these principles more readily understandable and applicable to system architects, designers, developers, and especially those who may not have formal security education or training.

Furthermore, it is important to note that these 18 security-oriented design principles are conceptual in nature and thus are used to inform a secure system design. They do not immediately provide a more secure system such as one might find with ''adding encryption'' or ''performing security penetration testing''. These principles contribute to designing and developing a complex system architecture that is fundamentally more secure and defensible from adverse cyber threats (malicious or non-malicious). In addition to being more secure, the resultant architecture should have higher levels of evidence-based trustworthy protection capabilities. Lastly, these principles are important for thinking and planning for the fluid nature of modern software-based systems such that the SoI can be more easily maintained and securely modified at less cost over the total system life cycle. This last point is particularly important as security improvements often become cost prohibitive in all but the most extreme situations.

1) CLEAR ABSTRACTIONS

Historically, the principle of clear abstractions has been applied to software functions and system interfaces to provide a consistent and intuitive view of the SoI's data – its data elements, data utilization, and data management. However, in a more general sense, clear abstractions means that the SoI should have a design that is obvious to others, functions that are distinct and well-defined, and information exchanges that are labelled and fully-characterized (*within reason*). Ideally,

the system design should be easily understandable and seemingly obvious to interpret by an engineer (for example).

Among the several security strategies and principles that clear abstractions supports, its contribution to the access control and isolation strategies is particularly evident given that it emphasizes the unambiguous definition of boundaries, behaviors, and relationships between subsystem elements, functions, users, dependencies, and data exchanges (i.e., all the SoI's significant objects and their associations). For example, information dependencies must be identified before access control rules can be defined.

While it is advantageous to initially define many abstractions, complete enumeration across the SoI's many users, subsystems, components, supporting/enabling systems, and various forms of data is difficult *(and arguably fleeting)*; thus, this principle must be constantly applied as the system's architecture, design, and implementation become further refined until the SoI is fully realized. Note, these abstractions also provide a detailed baseline for standardizing software and testing cybersecurity issues.

2) LEAST COMMON MECHANISM

The principle of least common mechanism seeks to reduce unnecessary duplication within the SoI. More specifically, if multiple components in a system require the same functionality, the desired functionality should be built into a single mechanism (physical or logical). For example, utilizing a single access control mechanism to implement a security policy can significantly reduce complexity across the SoI, as well as, reduce errors of omission and commission which may introduce vulnerabilities.

The least common mechanism security principle directly contributes to both access control and isolation strategies. Additionally, this principle supports reduced life cycle costs since there are not multiple instantiations of the same functionality to document and modify. On the other hand, taken to the extreme, least common mechanisms can result in a monoculture which contributes to reduced system resiliency where diversity helps to minimize the rapid propagation of an attack. Moreover, diversity provides alternative means of delivering required functionality so this principle has the potential to conflict with a system's resiliency goals [40], [41].

3) MODULARITY AND LAYERING

Modularity organizes and isolates functionality and related data into well-defined conceptual groupings, while layering orders the relationships between these groupings and their associated data flows. Note, we use the term ''groupings'' to refer to systems, system elements, components, or software objects with a systems engineering perspective rather than a narrower software engineering perspective as would typically be the case when discussing modularity and layering. While modularity is very closely related to clear abstractions, they are certainly different (e.g., a software object can be clearly defined but not modular; likewise, it can be modular but poorly defined). Confusion often occurs with these terms because a ''good software object'' is clearly defined, modular, and inherently layered. Perhaps, a helpful way to think about these principles is to consider that layering implies an ordered hierarchy of groupings (systems elements or objects), modularity implies ''plug-and-play'' of the groupings within the hierarchy, and clear abstractions implies that the hierarchy and its groupings are well-defined.

This principle is widely used in system development and especially modern software engineering practices, and as such, it contributes to each of the four security strategies and several principles. Of note, modularity and layering is particularly critical for supporting isolation and the resiliency strategy because the extent and impact of undesired behaviors can be more easily constrained (whether they be malicious or non-malicious). Lastly, it is worth mentioning that ''layering'' is not synonymous with ''defense in depth''; the former is focused on the efficient design of a system while the latter is typically focused on providing redundant and heterogeneous forms of protection.

4) ORDERED DEPENDENCIES (PARTIALLY)

In general, ordered dependencies refers to the logical (or hierarchical) arrangement of system elements (e.g., layers, modules, objects, etc.), and specifically, an ordering which functional calls, synchronization, and other dependencies are achieved through linearization or hierarchical design. Partially ordered dependencies simply refers to when the ordered dependency is mostly hierarchical (for example) but may require an additional non-hierarchical linkage. Ordered dependencies also seeks to minimize (or ideally, remove) circular dependencies.

Logically structuring dependencies (and minimizing them where possible) contributes to isolation between layers, increases understandability of the design, reduces system complexity, and facilitates testing and analysis. A system with partially ordered dependencies is also less likely to adversely impact associated functions and elements, thus contributing to survivability. Finally, ordering dependencies helps preserve trustworthiness by avoiding linkages between components with lower and higher trust levels.

5) EFFICIENTLY MEDIATED ACCESS

The efficiently mediated access principle ensures security mechanisms do not adversely hinder system performance to an unacceptable level. More generally, this principle seeks to ''optimize'' security and performance tradeoffs such that stakeholder systems security requirements (or goals) are met while the SoI performs its mission critical functionality. For example, security policies should be enforced at the lowest level possible with the most effective mechanism (physical or logical) within expressed constraints. In specific cases, the efficiently mediated access principle will overlap and result in the same outcome as the least common mechanism principle (but not as a general rule).

This principle directly contributes to the realization of the access control and isolation strategies, and it is related to

several other security principles such as least common mechanism, minimized sharing and reduced complexity. Of note, whereas verification and validation should be accounted for in the development of sound security principles these requirements often are not sufficiently addressed. As such, the deliberate application of this principle is important because analytically-based evidences are used to substantiate claims of trustworthiness, fulfillment of security objectives, and determination of risk.

6) MINIMIZED SHARING

The minimized sharing principle states that no resources should be shared between system components unless it is absolutely necessary to do so. Historically, this principle has been used to describe single-purpose hardware such as separate computer hard drives; however, the principle is generally applicable to system elements, software processes, interfaces, and humans. This principle directly supports the access control and isolation strategies by reducing the number of interactions between various system elements.

Restricting the sharing of resources has several benefits and effectively creates boundaries within the SoI to protect critical functions, simplify design, and streamline implementation. Separation can also be used to facilitate defense in depth solutions. However, while limiting shared resources is generally advantageous for security solutions, this principle may limit the resiliency solution trade space. Thus, the benefits of its application should be carefully considered when designing for security and resiliency.

7) REDUCED COMPLEXITY

The principle of reduced complexity implies that the system design should be as simple and small as possible. Clarity of design simplifies the understandability of the SoI, reduces total life cycle costs, and has several benefits especially when considering the development of secure systems. For example, reduced complexity directly contributes to the proper design and correct implementation of security and resiliency mechanisms. Moreover, that said mechanisms can be thoroughly tested and provide convincing evidences that the desired protection capability is achieved (i.e., verification and validation). It also contributes to identification of potential vulnerabilities which would have been obfuscated otherwise. Additionally, simpler designs can reduce the number of unnecessary dependencies which further reduces the system's attack surface.

Because of the additional insight gained through simplicity, this principle strongly contributes to nearly every design principle. However, if the application of this principle is taken too far, it may preclude the employment of isolation strategies, as well as, system redundancies which can negatively impact resiliency. Because of these competing dynamics, this principle merits careful attention.

8) SECURE EVOLVABILITY

The principle of secure evolvability implies that a system be developed to facilitate upgrades and maintenance activities in a secure fashion. This means that the SoI should be designed to easily and securely adapt to changes in its configuration, functionality, architecture, structure, interfaces, and interconnections. Given the fact that complex systems typically have life cycles spanning many years and face dynamic, constantly evolving threats, this principle is key to improving system security, resiliency, and survivability. Thus, this principle needs to account for the secure development and implementation of hardware and software upgrades, modifications, and patches during operations and sustainment.

Thus, this principle supports nearly all security strategies and principles throughout the SoI's life cycle. Focusing on secure evolvability early in the life cycle also has the potential to drive down engineering costs and facilitate fewer complex mitigations to future threats once the system has been deployed. On the other hand, it can adversely impact access control strategies if subsequent upgrades to the system are not carefully designed (and as a result, inadvertently introduce vulnerabilities into the SoI).

9) TRUSTED COMPONENTS (ELEMENTS)

The trusted components principle implies that a component must be at least as trustworthy as the components it supports. In more colloquial terms, this security principle simply implies that a chain is only as ''trustworthy'' as the ''weakest link'' in the chain. More formally, the trusted components principle implies that a component (or system element) must be trustworthy to at least a level commensurate with the security dependencies it supports. While originally conceived of for hardware components and devices, this principle is valid for the entire SoI and is applicable to component, elements, networks, functions, and other more subtle dependencies such as data and personnel which are often assumed to have high level of trustworthiness. This principle is foundational for the development (and operation) of assured systems because it enforces the requirement for valid evidences that support decisions of trustworthiness. Moreover, it ensures trust is not misplaced or inadvertently diminished throughout the development process.

Trusted components, and specifically the resulting trustworthy behaviors, enable the development of trustworthy secure systems such that desirable levels of trustworthiness can be achieved through a systematic approach. For example, how much testing is necessary to determine when an autonomous vehicle should be trusted to drive without a human operator? Returning to the security chain analogy, systems-level analysis of this principle enables the reasoning about and justification of the SoI's ''trust chain'' and highlights where trust is being hindered by less trustworthy ''chain links''. This principle is highly related to other trust and structurally-oriented principles including hierarchical trust and commensurate protection.

10) HIERARCHICAL TRUST

Building upon the principle of trusted components, hierarchical trust provides a logical basis for reasoning about

levels of trustworthiness when developing a system from several components. Hierarchical trust is especially pertinent with modern SoI, as developers consider the composition of numerous components, networks, and data at multiple levels of hierarchy each with differing levels of trustworthiness. Thus, hierarchical trust is an essential principle for reasoning about, and ultimately achieving, trustworthy secure systems.

Within the context of NIST SP 800-160, hierarchical trust supports the strategy of access control and to a lesser extent resiliency. It is worthwhile to note that this principle also depends on the correct implementation of several other design principles (more so than many other principles) because complex systems are composed of various components (and their dependencies) each of which imply differing levels of trustworthiness. In particular, hierarchical trust supports the principles of trusted components, hierarchical protection, and partially ordered dependencies which collectively provide evidences for reasoning about and justifying trustworthiness decisions.

11) COMMENSURATE PROTECTION

The principle of commensurate protection implies that the degree of protection provided for a component must be appropriate to its desired level trustworthiness. For example, as the trust placed in a component increases, the protection against unauthorized modification of said component should increase to the same degree. This principle is consistent with historic security defense approaches in that the most valuable items should be very well protected. Moreover, the term ''commensurate'' implies that the protection expenditures should correspond to the value of the items being protected. For example, the cost to defend the SoI should not exceed the total cost of the SoI. More practically, the cost to protect the SoI will be a percentage of the system lifecycle cost and must be justifiable to stakeholders.

This principle specifically contributes to the access mediation and defense in depth strategies, and builds on the principles of trusted components and hierarchical trust, as well as, several others to ensure that adequate trustworthiness is designed-in to the SoI. It also indirectly supports several other principles such as hierarchical trust, self-reliance and trusted communication. By deliberately focusing on the SoI's most critical components and functions, this principle ensures that supporting evidences are available for claims of trustworthiness.

12) HIERARCHICAL PROTECTION

The principle of hierarchical protection means that a component need not be protected from more trustworthy components. In addition to conventional physical components, this principle should be applied to software, information dependencies, and other system elements. It is akin to the principle of trusted components and hierarchical trust, but specifically addresses the aspect of unnecessarily over-protecting a component. In a similar way, this principle is akin to commensurate protection but applied where the subject component establishes the minimum required protection threshold for a complex system composed by multiple elements and/or components.

This principle supports all of the security strategies (to varying degrees) by protecting the SoI from components, functions, data, and even users with lower trustworthiness and reserving higher level privileges for more trusted entities. Regarding systems security design decisions, this principle is very important for guiding the application of architectural and component level principles to ensure security resources are not wasted on protections against higher trust level components.

13) MINIMIZED TRUSTED COMPONENTS

The principle of minimized trusted components implies that a system should not have extraneous trusted elements, components, data, or functions. Similarly to the foundational ''keep it simple'' engineering principle (described as ''reduced complexity'' in the NIST SP 800-160), the minimized trusted components principle suggests that the SoI should contain as few trustworthy components as possible. Thus, this principle directly impacts multiple facets of system development and especially total lifecycle costs.

This principle supports several security strategies by reducing the system's attack surface and improves the system's resiliency by reducing the SoI's likelihood of component-level failure. In particular, resiliency thinking often considers the possibility that systems and/or their components will occasionally be compromised, for example when facing a persistent intelligent adversary. Thus, minimizing the number of trusted components helps to mitigate the impact of those compromises, lowers assurance costs, and facilitates easier monitoring of a system's security posture [40]. However, minimizing the number of components may negatively restrict the SoI's design flexibility and collective resiliency capability.

14) LEAST PRIVILEGE

The principle of least privilege implies that a component should be allocated sufficient privileges to accomplish its specified function, and no more. While often described with respect to system users, when employed at the systems level, the principle is much more widely applicable to each system element, component, network, user, and more. Additionally, although the principle of least privilege is pervasive in computer security literature, it is often difficult to realize. For example, thorough testing can be accomplished to demonstrate a desired outcome occurs, however, it is much more difficult to demonstrate that undesired outcomes do not occur.

The least privilege principle directly supports access control, often with the outcome of logical or physical isolation to help minimize the impact of potential failures, corruption, misuse, and malicious activities. It also serves to reduce interdependencies, which simplifies component design, implementation, and analysis. Judicial application

supports resiliency and can improve survivability as an attacker's movements are limited when they are denied privilege escalation, a key tactic employed by advanced persistent threats [40].

15) MULTI-FACTOR PERMISSIONS

Formally presented as "Predicate Permission" in NIST SP 800-160, the principle of multi-factor permission requires that multiple authorizing entities (or operators) provide consent before a critical operation occurs. While traditionally thought of as two operators turning a key or offering a password, this principle should include authorization through multiple means to include human and system. Most commonly, multi-factor permission is granted through successive approval actions (i.e., a sequence of events occurring without error). Although slightly different, this concept can be loosely achieved through independent validation where the independent party can act as a second factor before granting permission or passing information.

This principle directly contributes to the access control strategy and defense in depth, while also enhancing survivability by protecting mission critical information, components, and processes. However, it is also important to consider that requiring multiple authentications also has the potential to negatively impact availability, survivability and/or resiliency. Lastly, the multi-factor permissions principle adds complexity into the design, so its application needs to be considered carefully within the systems security trade space.

16) SELF-RELIANCE

The principle of self-reliance means that systems and all its elements to include information and software should minimize their reliance on other systems, elements, or components. While this principle is specifically targeted at building trustworthiness, it is generally applicable for protecting the SoI as well. Applying the self-reliance principle can significantly reduce design and implementation complexity, increase testability, and improve system survivability. However, it is worth nothing that a SoI's ability to maintain situational awareness is often dependent upon the system's capability to monitor sub-systems, data feeds, and personnel interactions which may be negatively impacted with strict adherence to self-reliance.

This principle directly impacts the isolation strategy by minimizing both external and internal dependencies. When applied outside the SoI, this principle serves to reduce the system's attack surface which can reduce susceptibility to vulnerabilities inherent in external systems and network links. When applied within the SoI's architecture, it serves to protect critical functions and components through isolation.

17) SECURE COMPOSITION

The principle of secure composition means that the combination of system elements designed to enforce a security policy should result in a system that enforces said policy at least as well as the individual components do. Like many of the design principles, this principle is rather intuitive but problematic in implementation when considering modern systems with growing complexity and emergent behaviors. Note, when considering the point of multiple elements enforcing the same security policy, several principles towards reduced complexity should be considered.

This principle supports the access control and defense in depth security strategies by ensuring the 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 supports nearly all design principles. This principle should especially be considered when evaluating interactions between various components where the developer must work to identify and assess emergent behaviors and properties.

18) TRUSTED COMMUNICATION

The principle of trusted communications means that each communication channel (i.e., an interface, link, or network) must be trustworthy to a level commensurate with the security dependencies it supports. While seemingly redundant, this principle specifically considers the necessary protections for the SoI's communication links and ensures weaknesses are not introduced via communication channels. This is important because communication channels are inherently more susceptible than the SoI's internal elements. Moreover, system stakeholders often don't own the communication channels, thus additional scrutiny is required to ensure they have the same level of protection as the components they support.

This principle directly contributes to the access control and isolation strategies. Because of the preponderance of communication requirements in modern systems, the application of this principle impacts several other principles such as efficiently mediated access, trusted components, and hierarchical trust principles.

V. ANALYSIS RESULTS

Shown in Fig. 2, our analysis generated two clusters (outlined in red and blue, respectively) with a mean likeness value of 0.752 (on a scale from 0 to 1) during a 100-iteration run of the algorithm. By limiting the DSM optimization to elements with strong positive relationships (and controlling the above-mentioned cluster size and interaction strength parameters), the DSM optimization algorithm is able to provide concise, meaningful groupings.

The first cluster (shown in red) represents architectural considerations of a system, while the second cluster (shown in blue) represents considerations which collectively contribute to the trustworthiness of a system. While these results are dependent upon the mappings of Fig. 2, and ultimately the expertise of those interviewed, the DSM clustering software is provided online for users to investigate and update as desired. With a majority of the principles being design-oriented, roughly two thirds of the strongly positive relationships occur in the first cluster with 11 of the 18 principles.

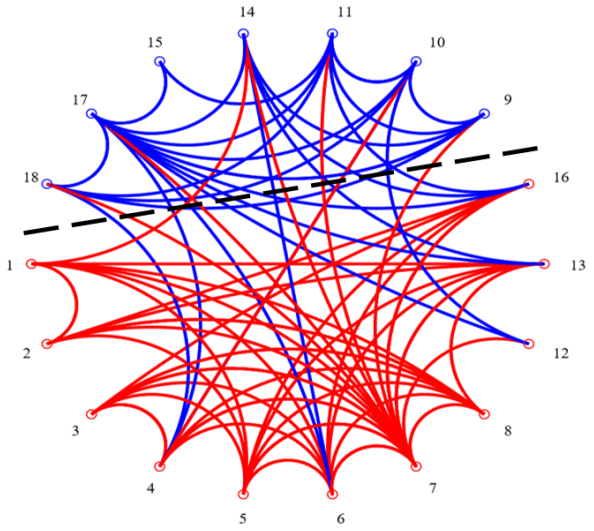

FIGURE 3. Relationship graph of the Design Structure Matrix (DSM) clustering optimization of interrelated systems security design principles with the names of design principles specified in Table 1. The trust related principles are shown in blue and the architectural related principles are shown in red. Note, directionality of the relationships is tabulated in the last two rows of Table 1 and detailed in the rows and columns of Fig. 2.

One notable exception is that three of the four strong positive elements supporting defense in depth are from the second cluster. The fact that the second cluster held a majority of the principles strongly related to defense in depth (Trusted Components, Commensurate Protection, and Secure Composition) is noteworthy in light of our earlier remark that "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" [10]. More specifically, poor implementation of these three principles detrimentally impacts the trustworthiness of a given system.

Fig. 3 provides an informative visual representation of the division and interconnectedness of the security principles among the two groupings with details provided in Table 1. In the relationship graph of Fig. 3, the color of each node represents which cluster it belongs to and the color of each edge identifies which cluster the relationship belongs to based on its originating node. Table 1 enumerates the total number of relationships (either supporting, supported, or two-way) and the number of relationships to the opposing cluster for each principle. This helps highlight several properties of our mapping and clustering. First, it provides a means of identifying the relational density of a given principle. For instance, principle 7 shows the strongest inter-relationship mapping (with a value of 15), while principle 15 has the weakest (being linked to only 2 others). While all 18 design principles may be important to the security of a system, this ranking can serve as an indicator of the relative criticality of each principle. Second, links crossing the clustering divide (shown as a dashed line) and particularly the principles with higher "# to Other Cluster" counts contribute to both clusters. In particular, clusters 12 and 17 exhibit a relatively high number of links to the opposing cluster, so they may warrant further consideration in both clusters (architectural considerations and trustworthiness).

A. INTERPRETATION

Given the two groupings, we now evaluate them for actionable insights to assist those charged with executing SSE roles and responsibilities. Beginning with a cursory inspection of the first cluster (shown in the red outline of Fig. 2), security principles associated with interfaces, boundaries, data flows, architecture, and other design-based concerns are identified. For instance, the clarity and simplicity of interfaces, data elements, and trusted components is promoted through the application of three design principles: clear abstractions, reduced complexity, and minimize trusted components. Likewise, least common mechanism, minimized sharing, and efficiently mediated access place emphasis on designs involving the fewest number of common mechanisms and shared resources to accomplish a specified function.

Longstanding engineering tenets such as modularity, layering, ordered dependencies, and hierarchical protection are concerned with essential system design decisions such as logical organization, structure, data flows, and the orderliness of function calls. Self-reliance helps to bound the design and engineering effort to minimize security, resiliency, and protection requirements which directly impact the cost and feasibility of proposed security solutions. Lastly, secure evolvability considers the application of these principles over the SoI's life cycle as it is transformed.

With respect to the system life cycle, the first cluster of principles is particularly beneficial for consideration during the earliest phases of a SoI's development. Specifically, these architectural-based principles should be considered during the inception and concept phases through requirements definition and design when the solution trade space is widest [24]. For example, effectively application of these principles can improve system security and resiliency in a cost-effective manner. Conversely, rigorous application of these architectural principles to a pre-determined or fixed physical architecture is of significantly less value.

Continuing our examination to the second cluster (shown in the blue outline of Fig. 2), this group of principles is strongly associated with the notion of trust and how the security engineer defines and ensures appropriate levels of trustworthiness are achieved. Since trust is used to form the foundation of how security analysts reason about the correctness of the system's design to its stated specification, it follows that these principles should be considered during the design phases (i.e., concept and development). Moreover, these principles need to be reconsidered throughout the system life cycle to ensure the SoI is properly operated and maintained in accordance with its intended purpose and emerging stakeholder's security needs. It is important to highlight that the SoI's required level of trustworthiness is not static, as its relative criticality to an organization (or mission) often evolves over time. Thus, the system's

specified degree of trustworthiness and associated protection mechanisms should be periodically re-evaluated, especially as major system modifications occur.

Regarding the specific trust-oriented principles, trusted components assure that the SoI is developed, fielded, and maintained with components that are determined to be trustworthy. Likewise, hierarchical trust and trusted communication ensure dependencies between trusted components and communication mechanisms are correctly accounted for. These principles take on increasing importance as changes to one component often impact the trustworthiness of other components which rely upon it (including the SoI itself). These trust principles are typically employed in conjunction with the principle of commensurate protection which collectively assure that the components (and combinations thereof) are protected to a level appropriate for the security dependencies they support. As a common means for securing critical data and operations, multi-factor permissions need to be considered during design activities, and not merely costly add-on modifications.

Similarly, secure composition seeks to maintain the trustworthiness of a given systems-of-system's collective security policy implementation as modifications can cause unexpected or emergent results as the SoI changes over time and/or is used differently than expected. While an important consideration during initial design, the principle of least privilege is critical during operations and should be re-validated from time to time, lest excessive permissions begin to aggregate in a system.

Finally, the relative importance of these trust-oriented design principles cannot be overstated as it is common to place unmerited trust in components that are without evidences to support their claims of trustworthiness. Likewise, while the importance of these security principles is often emphasized during early design efforts, their importance across the entire system life cycle is less so, where planned and unplanned repairs, modifications, replacements, and upgrades can undermine critical trust relationships and protection measures.

B. NOTIONAL CASE STUDY EXAMPLE

To demonstrate how a systems security engineer can make use of the proposed principle groupings, we briefly consider the development of a notional intra-theater ballistic missile defense system shown in Fig. 4 [42]. Early consideration of the trust and architectural principle groupings is helpful for thinking about the entire system life cycle for a non-trivial SoI in order to meet both security and resiliency stakeholder needs. For example, Fig. 5 provides a general overview of how the developer's level of attention to these two principle groupings might increase and decrease throughout the SoI's life cycle. Detailed in the next five Sections, we describe tradeoffs between these groupings for developing secure and resilient systems. Additionally, we provide some commentary as to how individual principles may be applied during each phase of the life cycle.

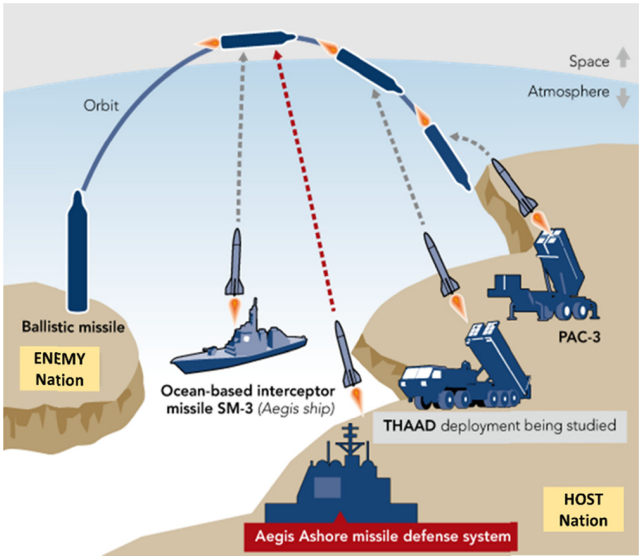

FIGURE 4. Notional intra-theater ballistic missile defense system for discussing the application of security design principle groupings throughout the system life cycle. Adapted from [42].

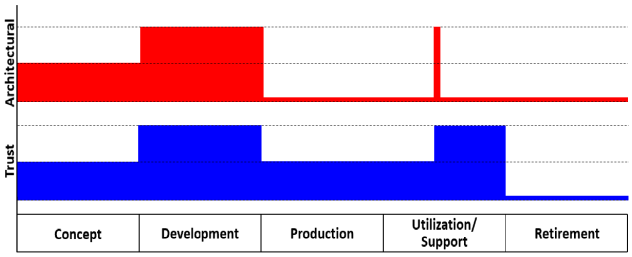

FIGURE 5. Notional application of the security design principle groupings throughout the system life cycle.

1) CONCEPT PHASE

Beginning with the concept phase, the architectural design principles are considered as a means for identifying, defining, considering, analyzing, and ultimately realizing a system that meets security and resiliency requirements for the desired capability. In this phase, it is prudent to consider principles such as Clear Abstractions to properly identify, define, decompose, and allocate the SoI's desired functionally (i.e., mission essential behaviors). Likewise, the principle of Modularity and Layering affects the high-level design of the system and directly contributes to security and resiliency decisions such as required data flows and what classification of data each element of the SoI will need to process and store. In this example, this system is to be highly trustworthy; thus, the trust-related principles should also be initially considered. Ideally, each principle should be systematically assessed for applicability to provide evidences for trustworthiness of mission essential or safety critical systems.

We would also like to highlight that we believe the key to developing secure and resilient systems is to fully understand their intended purpose (i.e., its desired capability) and proposed operational environment. In our experience, we've found that the System-Theoretic Process Analysis for

TABLE 1. Relationship counts per principle.

Principle	1. Clear Abstractions	2. Least Common Mechanism	3. Modularity and Layering	4. Ordered Dependencies	5. Efficiently Mediated Access	6. Minimized Sharing	7. Reduced Complexity	8. Secure Evolvability	9. Trusted Components	10. Hierarchical Trust	11. Commensurate Protection	12. Hierarchical Protection	13. Minimize Trusted Components	14. Least Privilege	15. Multi-Factor Permissions	16. Self-Reliance	17. Secure Composition	18. Trusted Communication
Total # of Relationships	7	6	6	7	8	10	15	8	6	6	7	3	8	7	2	8	12	6
# to Architectural Cluster	**6**	**6**	**6**	**5**	**8**	**8**	**10**	**8**	1	**4**	**5**	1	**6**	3	**2**	**5**	**11**	1
# to Trust Cluster	1	0	0	2	0	2	5	0	**5**	2	2	**2**	2	**4**	0	3	1	**5**
# of Supported Principles (Across each row)	7	2	4	3	3	3	13	6	3	3	3	0	4	4	1	5	10	4
# of Contributing Principles (Down each Column)	0	5	5	7	6	8	8	4	4	5	5	3	5	3	1	4	3	2

Security (STPA-Sec) is a simple and straightforward approach for understanding and framing the SoI's purpose at a conceptual level which helps to elicit meaningful security and resiliency requirements within a mission-informed context [28].

2) DEVELOPMENT PHASE

During the development phase, scrutiny of the system's detailed design places increased emphasis on both groups of principles. This is because the SSE principles are fundamental in designing secure and resilient systems. Moreover, unintentional security design omissions often create critical problems later in the life cycle. For example, fixing vulnerabilities in a fielded system is often very costly, if not entirely prohibitive. Thus, systems security engineers should work diligently to avoid foreseeable weaknesses, diagnosable faults, and potential failures modes. Moreover, these engineers should also seek to conceptualize and develop systems that are easier for users to operate and sustain which can be achieved by applying the design principles not only for security but more generally to all aspects of the SoI.

In exercising the application of the security-oriented design principles during this case study (and a second one to be described in a future paper), we realized it is easy to become ''stuck'' in the process. This is because tradeoffs are inherent in complex system development as even DARPA's most secure systems to date are forced to make key security concessions [25]. With respect to the design principles, we believe this is because it is generally desirable to maximally employ each design principle, yet several conflicts and tradeoffs exist (as detailed in Figs. 2 and 3). For example, additional communication links may increase resiliency yet hinder security. Consequently, in order to achieve the SoI's desired protection capability, as well as, control long-term cost and maintainability, additional assistance is needed in efficiently applying these principles.

In Table 2, we provide a helpful listing of several engineering considerations to aid system developers and security practitioners in their SSE developmental effort. Specifically, we want to aid in the application of the NIST SP 800-160 Vol. 1 security design principles, so we've captured various practitioner-oriented questions and insights that have come to light during our case study. For example, consider the two ''most connected'' principles from Fig. 2–#7. Reduced Complexity and #17. Secure Composition from the architectural and trust groupings, respectively. The principle of reduced complexity is architecturally significant and contributes strongly towards designing defensible and resilient systems where eliminating excess interfaces and dependencies reduces the SoI's attack surface. Likewise, the secure composition principle is important for building systems with a justifiable trust level as each element of the SoI contributes to or diminishes from the SoI's resultant trustworthiness level. Following the questions and considerations presented in Table 2 provides a systematic way to address the application of the NIST SP 800-160 Vol. 1 principles that is relatively easy to follow and not too confusing.

While Table 2 is particularly focused on the development phase of the system life cycle, these principles are generally applicable to several SSE processes, activities, and tasks across the entire life cycle; however, please note these questions must not be blindly followed without understanding the SoI's mission and/or business processes.

3) PRODUCTION PHASE

Once the production phase is reached, the architectural-based principles are deemphasized. Presumably by this time the principles have served their purpose to help define a system that, when operational, will be more secure and resilient (this is especially true when engineering rigor and systems analysis is applied early in the life cycle). At the production phase, the engineering effort is typically focused on ensuring that

TABLE 2. Systems security design principle engineering considerations.

Principle Name	Simplified Descriptions	Considerations and Guidance
1. Clear Abstractions	A system should have simple, well-defined interfaces and functions.	• Is the system design readily understandable to a third party? • Are the SoI's functions and subsystems obvious to an independent third party? • Are the data elements precisely named and well-defined? • Are there defined interfaces for every information input, output, and control?
2. Least Common Mechanism	If multiple components in a system require the same functionality, the desired functionality should be built into a single mechanism (logical or physical).	• Does the SoI have security features which are repeated across the SoI's functions and subsystems? • Can the security functionality be consolidated (where appropriate)?
3. Modularity and Layering	Organize and order functionality and related data flows between entities.	• Is the system design readily understandable to a third party? • Are the SoI's functions and subsystems obvious to an independent third party? • Separate and eliminate dependencies where possible. • Is there a defined interface for every information input/output? • Are the data elements precisely named and well-defined?
4. Ordered Dependencies	Logically arrange modules and layers such that linear (or hierarchical) function calls, synchronization, and other dependencies are achieved.	• Are security dependencies identified? • Are security dependencies one way – to prevent circular dependencies?
5. Efficiently Mediated Access	Security enforcement mechanisms (logical or physical) should not unnecessarily impact the SoI's performance/purpose.	• What is the applicable security policy? • Are the SoI's security requirements and constraints specified? • What means are available for meeting these policies, requirements, and constraints?
6. Minimized Sharing	Only share between components when absolutely necessary.	• Reduce and/or eliminate resource sharing as much as possible.
7. Reduced Complexity	Develop a simple and small system.	• Is the system design readily understandable to a third party? • Have unnecessary requirements been eliminated? • Is it possible to reduce any functions, subsystems, dependencies, or interfaces?
8. Secure Evolvability	Plan for secure maintenance and sustainment activites over the SoI's life cycle.	• Has maintainability been designed into the system? • Have user-level security monitoring capabilities been designed into the system? • Have considerations been made for human learning and process changes? • Have dependencies on legacy systems been minimized?
9. Trusted Components	A component must be trustworthy to at least a level commensurate with the security dependencies it supports.	• Identify the components that transmit, receive, and store critical information. • Ensure said components operate at the highest information classification level. • Are the SoI's functions and subsystems obvious to an independent third party? • Are the data elements precisely named and well-defined? • Is there a defined interface for every information input/output?
10. Hierarchical Trust	An ordered hierarchy should be used when composing a system from multiple components.	• Is there a defined interface for every information input/output? • Are the data elements precisely named and well-defined? • Does the security hierachy build appropriately?
11. Commensurate Protection	The degree of security provided should be commensurate with an asset's value.	• Is the security requirement appropriate for the asset being protected? • Are high trust assets and information protected appropriately? • Is too much protection being applied to a low value asset? • Is too little protection being applied to high value assets?
12. Hierarchical Protection	It is not necessary to protect a less secure component from a more secure component.	• Does a system-level security hierarchy exist? • Are there any circular relationships?
13. Minimize Trusted Components	A system should have a limited set of trusted elements, components, data, and functions.	• Eliminate high security functions, data, and components whenever possible.
14. Least Privilege	Each component should only have sufficient privilege to accomplish its specified function.	• Are the SoI's privileges identified, defined, and captured in a single location? • For each function, subsystem, or component is the principle of minimum access employed?
15. Multi-Factor Permissions	Highly critical operations and data should require multiple authorization steps.	• Do mission critical activties and data require multi-factor authorization? • Do these multi-factor permissions inhibit mission execution?
16. Self-Reliance	Minimize dependencies on other systems, components, infrastructure, and data.	• Are internal dependencies reduced? • Are external dependencies reduced?
17. Secure Composition	Combining secure components or functions should not reduce their collective level of trustworthiness.	• Has the SoI's assembly reduced the trustworthiness of the security capability? • Has the SoI's assembly reduced the trustworthiness of the resiliency capability?
18. Trusted Communication	Communication links and networks must be secured to the same level as the components and information that they carry.	• Are the communication requirements known and well understood? • Are secure communication channels properly identified in an architectural document (conceptual and/or physical)? • Are the secure communication channels adaquetely secured? • Have unnecessary high security information networks and communication links been eliminated or minimized?

the SoI is built in accordance with the design specifications with additional consideration for the trustworthiness of the system such that appropriate levels of trustworthy security and resiliency are maintained throughout implementation. For example, nation-state adversaries have demonstrated the ability to inject vulnerabilities during each phase of the development life cycle [1], so it is important that the trust-related principles (and the trustworthiness of the components produced) are continually monitored and assessed. Likewise, suitable training must be created (and successfully delivered) for operational personnel to be able to detect and respond when the SoI is under cyber attack. Unfortunately, this important human-system integration aspect of cyber systems is often neglected for more tangible hardware and software issues.

4) UTILIZATION/SUPPORT PHASE

The initial transition to the utilization phase is unlikely to merit any changes to the developer's level of effort as there is little extra to be gained from the design and architectural principles once the SoI is fielded; however, the trust principles will always require a degree of constant attention to ensure the system is being operated as designed. During the support phase (i.e., whenever modifications occur whether small or large), it's incumbent on the user and developers to pay careful attention to changes in the SoI, its operational environment, and its performance to include any supporting personnel, processes, and technology to properly understand the SoI's trustworthiness.

For example, during operations those responsible for assuring the security of the SoI should remain vigilant in monitoring how the system is being used to understand if changes to its usage and/or the threats it faces require modifications to the SoI. To illustrate this point, suppose that during the utilization/support phase there are increasing tensions in a given geopolitical region with a nuclear-armed adversary whom possess inter-continental ballistic missiles. Accordingly, the military elects to modify the system by installing a space-based early warning detection capability and utilize the existing SoI to relay indications of enemy launches. Moreover, these intra-theater notifications are to be integrated into a national missile defense system via the satellite ground station. What was once merely a theater-specific ballistic missile warning system is now going to be part of a larger system-of-systems which supports a critical, strategic-level military capability. As indicated by the utilization/support spike in Fig. 5, the system developer should once again revisit the architectural and trust principles to ensure the necessary modifications and resulting system composition still meets the SoI's trustworthiness requirements (which should be commensurate with its new role).

For instance, the developer may need to determine if the communications channels are sufficiently protected for this new mission (i.e., Trusted Communication) or address the vulnerability of commercial off-the-shelf elements in the supply chain (i.e., Trusted Components). At this point the system developer is presumably thankful that consideration was given for the Secure Evolution of the SoI. Once the system is modified, the architectural-based principles are de-emphasized once again, but the trust-related principles should be continually monitored to assure the trustworthiness of the system (which is now at a higher level due to the SoI's relative increase in strategic importance to its key stakeholder–the military).

5) RETIREMENT PHASE

Once the system has reached the end of its useful life, it should be properly decommissioned with respect to both systems security and trust. For example, ensuring no sensitive configuration information remains in the SoI prior to its disposal where an adversary may gain operational insight. In another example, proprietary interfaces should be destroyed or removed to prevent ease of access.

VI. CONCLUSION

This article is part of a series of works which aims to assist developers, owners, and operators in understanding and achieving a rigorous, yet cost-effective SSE approach. This work uniquely provides an analysis of the NIST SP 800-160 Vol. 1 security design principles with a detailed mapping and analysis of conceptual security strategies to design principles that can be more effectively designed-for, built-in, and tested to meet security and resiliency objectives. Insights regarding the interrelationships amongst the 18 design principles are described along with practical guidance on when and how they can be applied to more effectively perform SSE activities.

While attempting to systematically apply the design principles to both the case study described within and an additional autonomous vehicle example (not described), we found it difficult at times to differentiate between the various principles. We also found redundancy between the design principles, especially as we abstracted up the principles to the systems-level from their original lower-level, computer security roots. This is not a criticism, per se, but merely a finding that leads us to believe that a smaller set of systems security design principles is probably likely. Additionally, in considering the application of these security design principles to an autonomous system, we found the principles had limited applicability to this growing area of interest. Again, we state this not as a criticism – as the original design principles are not specifically for autonomous systems – but merely to suggest that design principles for trustworthy autonomous cyber-physical systems would be an interesting area to study.

Our future work includes: 1. Identifying associated technical performance measurements for these design principles; 2. Analyzing the application of these principles through modeling and simulation; and 3. Further understanding the applicability of these principles for autonomous systems.

ACKNOWLEDGMENT

The authors would like to thank Air Force Research Laboratory, Space Vehicles Directorate, Kirtland AFB, NM and Michael A. McEvilley, co-author of NIST SP 800-160 Systems Security Engineering for their support in this research effort. They would also like to thank the participants of NATO panel IST-164 for attempting to apply these design principles to an autonomous SoI. The lessons learned from the NATO working group are very helpful towards refining the key ideas presented in this paper.

DISCLAIMER

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U. S. Air Force, the Department of Defense, or the U.S. Government.

REFERENCES

[1] J. R. Gosler and L. Von Thaer, *Task Force Report: Resilient Military Systems and the Advanced Cyber Threat*, Dept. Defense Bus. Board, Washington, DC, USA, 2013.

[2] K. Baldwin, J. F. Miller, P. R. Popick, and J. Goodnight, "The United States department of defense revitalization of system security engineering through program protection," in *Proc. IEEE Int. Syst. Conf. SysCon*, Mar. 2012, pp. 1–7.

[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*. Santa Monica, CA, USA: RAND Corporation, 2015.

[4] *A Path Towards Cyber Resilient and Secure Systems*, NDIA Syst. Eng. Division, Virginia, VA, USA, 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 Secur. Privacy*, vol. 14, no. 4, pp. 76–80, Jul./Aug. 2016.

[6] R. Ross, M. McEvilley, and J. C. Oren, *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. Gaithersburg, MD, USA: NIST, 2018.

[7] R. Ross, R. Graubart, D. Bodeau, and R. McQuaid, *Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems (DRAFT*. Gaithersburg, MD, USA: NIST, 2018.

[8] *IEC/IEEE International Standard-Systems and Software Engineering–System Life Cycle Processes*, 1st ed., ISO/IEC/IEEE International Standard, ISO/IEC/IEEE, 15288-2015, 2015.

[9] D. J. Bodeau, R. D. Graubart, J. Picciotto, and R. McQuaid, *Cyber Resiliency Engineering Framework*. Bedford, MA, USA: MITRE Corporation, 2011.

[10] P. M. Beach, L. O. Mailloux, and M. T. Span, "Examination of security design principles from NIST SP 800-160," in *Proc. Annu. IEEE Int. Syst. Conf. (SysCon)*, Vancouver, BC, Canada, Apr. 2018, pp. 1–8.

[11] C. Irvine and T. D. Nguyen, "Educating the systems security engineer's apprentice," *IEEE Secur. Privacy*, vol. 8, no. 4, pp. 58–61, Jul. 2010.

[12] J. Bayuk, "Systems security engineering," *IEEE Secur. Privacy*, vol. 9, no. 2, pp. 72–74, Mar./Apr. 2011.

[13] K. Baldwin, "System security engineering: A critical discipline of systems engineering," *Insight*, vol. 12, no. 2, pp. 11–13, 2009.

[14] L. Masso and B. Wilson, "Management initiatives to integrate systems and security engineering," *Insight*, vol. 16, no. 2, pp. 10–12, 2013.

[15] J. C. Oren, "What does a systems security engineer do and why do systems engineers care?" *Insight*, vol. 16, no. 2, pp. 16–18, 2013.

[16] R. Anderson, *Security Engineering*, 2nd ed. Indianapolis, IN, USA: Wiley, 2008.

[17] J. Bayukm, D. Barnabe, J. Goodnight, D. Hamilton, and B. Horowitz, *Systems Security Engineering Final Technical Report*. Hoboken, NJ, USA: Stevens Institute, 2010.

[18] *Military Handbook 1785. System Security Engineering Program Management Requirements*, Dept. Defense, Washington, DC, USA, 1989.

[19] N. S. Agency. (Nov. 2016). *National Centers of Academic Excellence*. Accessed: May 30, 2017].[Online]. Available: https://www.nsa.gov/resources/educators/centers-academic-excellence/cyber-operations/requirements.shtml

[20] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. Hoboken, NJ, USA: Wiley, 2013.

[21] Defense Acquisition University. (May 2017). *System Survivability Key Performance Parameter*. Accessed: Jun. 1, 2017. [Online]. Available: https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=626ace5f-fc1f-4638-9321-fe4345451558

[22] T. V. Benzel, C. E. Irvine, T. E. Levin, G. Bhaskara, T. D. Nguyen, and P. C. Clark, "Design principles for security," Dept. Comput. Sci., Nav. Postgraduate School, Monterey, CA, USA, Tech. Rep. NPS-CS-05-010, 2005.

[23] X. Sun, P. Liu, and A. Singhal, "Toward cyberresiliency in the context of cloud computing [resilient security]," *IEEE Secur. Privacy*, vol. 16, no. 6, pp. 71–75, Nov./Dec. 2018.

[24] *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 4*, Int. Council Syst. Eng., San Diego, CA, USA, 2014.

[25] D. Cofer, A. Gacek, J. Backes, M. W. Whalen, L. Pike, A. Foltzer, M. Podhradsky, G. Klein, I. Kuz, J. Andronick, G. Heiser, and D. Stuart, "A formal approach to constructing secure air vehicle software," *Computer*, vol. 51, no. 11, pp. 14–23, Nov. 2018.

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

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

[28] M. Span, L. O. Mailloux, R. F. Mills, and W. Young, "Conceptual systems security requirements analysis: Aerial refueling case study," *IEEE Access*, vol. 6, pp. 46668–46682, 2018.

[29] E. Crawley, B. Cameron, and D. Selva, *System Architecture: Strategy and Product Development for Complex Systems*. Upper Saddle River, NJ, USA: Prentice-Hall Press, 2015.

[30] D. V. Steward, "The design structure system: A method for managing the design of complex systems," *IEEE Trans. Eng. Manage.*, vol. EM–28, no. 3, pp. 71–74, Aug. 1981.

[31] R. E. Thebeau, *Knowledge Management of System Interfaces and Interactions From Product Development Processes*. Cambridge, MA, USA: Massachusetts Institute of Technology, 2001.

[32] F. Borjesson and K. Hölttä-Otto, "Improved clustering algorithm for design structure matrix,"in *Proc. ASME Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, Aug. 2012, pp. 921–930.

[33] P. G. Neumann, "Reflections on system trustworthiness," *Adv. Comput.*, vol. 70, pp. 269–310, Jan. 2007.

[34] A. Avižienis, "A visit to the jungle of terminology," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops*, Jun. 2017, pp. 149–152.

[35] National Defense Industrial Association. (2017). *NDIA Systems Engineering Cyber Resilient & Secure Weapon System Summit 2017*. Accessed: May 5, 2017. [Online]. Available: http://www.ndia.org/events/2017/4/18/ndia-systems-engineering-cyber-resilient-and-secure-weapon-system-summit-2017

[36] O. Sokolsky and N. Bezzo, "Resiliency in cyber-physical systems [Guest Editors' Introduction]," *Computer*, vol. 51, no. 11, pp. 10–12, Nov. 2018.

[37] J. Gao, B. Barzel, and A. L. Barabási, "Universal resilience patterns in complex networks," *Nature*, vol. 530, no. 7590, pp. 307–312, Feb. 2016.

[38] D. J. Bodeau, R. D. Graubart, J. Picciotto, and R. McQuaid, *Cyber Resiliency Engineering Framework*. Bedford, MA, USA: MITRE Corporation, 2011.

[39] J. Hughes and G. Cybenko, "Three tenets for secure cyber-physical system design and assessment," *Proc. SPIE*, vol. 9097, Jun. 2014, Art. no. 90970A-1.

[40] D. Bodeau and R. Graubart, *Cyber Resiliency Design Principles*. Bedford, MA, USA: MITRE, 2017.

[41] J. Knight, J. Davidson, A. Nguyen-Tuong, J. Hiser, and M. Co, "Diversity in Cybersecurity," *Computer*, vol. 49, no. 4, pp. 94–98, Apr. 2016.

[42] K. Sakai. (Aug. 17, 2017). Japan to Deploy New Land-Based Missile Defense System. Nikkei Asian Review. Accessed: Mar. 29, 2019. [Online]. Available: https://asia.nikkei.com/Politics/Japan-to-deploy-new-land-based-missile-defense-system

PAUL M. BEACH received the B.S. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, USA, in 2006, and the M.S. degree in leadership & information technology from Duquesne University, Pittsburgh, PA, USA, in 2009. He is currently pursuing the Ph.D. degree in systems engineering. He is also a Commissioned Officer in the United States Air Force. His research interests include systems security engineering, cybersecurity, and dynamic data-driven application systems.

LOGAN O. MAILLOUX (M'12–SM'18) received the B.S. degree in computer engineering from Lawrence Technological University, Southfield, MI, USA, in 2002, the M.S. degree in systems engineering, in 2008, and the Ph.D. degree in systems engineering from the Air Force Institute of Technology, Dayton, OH, USA, in 2015. He serves as a Commissioned Officer in the United States Air Force. His research interests include system security engineering, cyber-physical systems, and quantum cybersecurity technologies.

BRENT T. LANGHALS received the B.S. degree from the US Air Force Academy, in 1995, the M.S. degree from the Air Force Institute of Technology, in 2001, and the Ph.D. degree in management information systems from the University of Arizona, in 2011. He is currently an Assistant Professor with the Air Force Institute of Technology, Wright-Patterson AFB OH. His research interests include big data, database/data storage, data analytics, dynamic data driven application systems, and cyber operations.

ROBERT F. MILLS (M'96–SM'08) received the B.S.E.E. degree from Montana State University, in 1983, the M.S.E.E. degree from the Air Force Institute of Technology, in 1987, and the Ph.D. degree in electrical engineering from the University of Kansas, in 1994. He is currently a Professor of electrical engineering with the Air Force Institute of Technology, Wright-Patterson AFB, OH, USA. His research interests include communications systems, electronic warfare, systems engineering, and cyber-physical systems security. He is a member of the Tau Beta Pi and Eta Kappa Nu.