Received June 19, 2018, accepted July 30, 2018, date of publication August 16, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2865736

Conceptual Systems Security Requirements Analysis: Aerial Refueling Case Study

MARTIN SPAN, III,[1], (Member, IEEE), LOGAN O. MAILLOUX[2], (Senior Member, IEEE), ROBERT F. MILLS[2], (Senior Member, IEEE), AND WILLIAM YOUNG, JR.,[3]

[1]United States Air Force Academy, Colorado Springs, CO 80841, USA
[2]Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, USA
[3]53rd Electronic Warfare Group, Eglin AFB, FL 32542, USA

Corresponding author: Martin Span, III (martin.span.1@us.af.mil)

This work was supported in part by the United States Air Force Institute of Technology Center for Cyberspace Research, Wright Patterson Air Force Base, OH, USA, and in part by the United States Air Force Research Laboratory, Space Vehicles Directorate, Kirtland Air Force Base, NM, USA.

ABSTRACT In today's highly interconnected and technology-reliant environment, cybersecurity is no longer limited to traditional computer systems and IT networks, as a number of highly publicized attacks have occurred against complex cyber-physical systems such as automobiles and airplanes. While numerous vulnerability analysis and architecture analysis approaches are in use, these approaches are often focused on realized systems with limited solution space. A more effective approach for understanding security and resiliency requirements early in the system development is needed. One such approach, system-theoretic process analysis for security (STPA-Sec), addresses the cyber-physical security problem from a systems viewpoint at the conceptual stage when the solution trade-space is largest rather than merely examining components and adding protections during production, operation, or sustainment. This paper uniquely provides a detailed and independent evaluation of STPA-Sec's utility for eliciting, defining, and understanding security and resiliency requirements for a notional next generation aerial refueling platform.

INDEX TERMS Cybersecurity, requirements engineering, security, security engineering, systems engineering, systems security engineering.

I. INTRODUCTION

The cybersecurity threat is one of the most serious challenges in the 21st century. Over the past decade, attacks have grown considerably in frequency and complexity, and it is now commonplace to hear of widespread attacks against personal computers, webservers and services, Internet of Things (IoT) devices, major retailers, and even critical government databases. Moreover, the security of Cyber-Physical Systems (CPS) is becoming increasingly important as these devices take on central roles in nearly every aspect of modern life. Previously, CPS such as automobiles and airplanes were complicated but not interactively complex. Security (and safety) is now best treated as an emergent property of CPS, where their software and real-time networks require previously isolated components to continuously interact to accomplish the system goal or purpose. For example, the 2017 Ford F-150, a fairly common vehicle in the United States, has over 150 million lines of code distributed across dozens of computing devices with software providing essential functionality [1]. Moreover, adversaries are challenging traditional assumptions that CPS are secure due to their relative isolation and uniqueness with recent examples including the widely publicized hacking of a Jeep Cherokee [2] and a commercial airliner [3], along with comprehensive reports of vehicular attack paths [4].

In light of these growing threats, it is critical for security professionals to have appropriate tools and techniques for performing Systems Security Engineering (SSE). For example, the United States Department of Defense (U.S. DoD) which historically values systems security, has made several recent changes to expand traditional IT-focused security approaches and mandate security assessments for cyber-physical weapon systems [5]. These policies dictate that acquisition programs integrate security efforts into existing systems engineering processes and work to ensure security considerations hold equal footing with other requirements and design trade-offs. It is not easy to understand what constitutes a ''secure'' system nor how to define security requirements or testable criteria, the foundation from which analysis and evidences are used to determine whether a system is ''secure.''

The challenge of cybersecurity is a ''wicked problem'' where the problem is twofold: first the problem itself must be defined. Then the solution or actions required to get from 'as-is' to 'to-be' must be determined. Specific to security, this wicked problem requires engineers to secure systems via an unknown solution and defend against an unknown evolving threat. Nested, interactive complexity and the socio-technical aspects make cybersecurity a wicked problem [6]. Traditional security approaches are typically conducted at a component level and the results aggregated together into a system analysis. This analysis assumes security is composable and fails to capture emergent properties of the system that arise from complex interactions. There are many current approaches to systems security and vulnerability analysis, with varying levels of success. For a more detailed survey of approaches relevant to weapons systems cybersecurity, see the authors' prior work [23]. Additionally, the motivation for this research is expanded in [7]. This research addresses weapon system security through executing a conceptual security analysis in a case study of interest to the United States Air Force, USAF.

This paper has three main goals: First, it provides a complete and thorough example of STPA-Sec for a complex aerial refueling system. Second, it offers recommendations for future practitioners. Third it demonstrates the utility of STPA-Sec for complex Major Weapons System (MWS) security analysis. This work presents a background of STPA-Sec in Section II. Section III provides an introduction to the case study and presents a tailored STPA-Sec approach. Sections IV-VI detail the steps and analysis performed for each phase of an STPA-Sec analysis. First, the purpose of the phase will be introduced, followed by a description of the steps. The case study example is presented along with rationale to assist a practitioner in accomplishing an STPA-Sec analysis for their complex System of Interest (SoI). Section VII provides an assessment of STPA-Sec's utility, and Section VIII provides a brief summary and conclusion.

II. BACKGROUND

STPA-Sec is a promising methodology for developing secure systems and performing a system security analysis. STPA-Sec applies a system engineering approach providing engineers the largest trade space for developing secure solutions. STPA-Sec elevates the security problem from guarding the system against all potential attack paths to the higher-level problem of assuring the system's critical functions [8]. Because STPA is a top-down, system engineering approach to system safety and security, it can be used early in the system development process to generate high-level safety and security requirements and constraints. These high-level requirements can be refined using STPA to guide the system design process and generate detailed safety and security requirements for individual components [9].

STPA-Sec is an extension of the systems safety focused STPA. The theory behind STPA is Leveson's Systems-Theoretic Accident Model and Process (STAMP) [10]. This work has been well received within the safety and systems engineering community [11]. It is founded on systems theory, analyzing the system as a whole rather than a sum of the parts to capture emergent properties common in complex systems. It asserts that safety and security are emergent properties resulting from relationships among the parts of the system. STAMP and STPA define the safety problem as a control problem and leverage control theory to design an effective control structure that reduces or eliminates adverse events [12]. STPA-Sec extends this methodology from the safety domain to the security domain and has been shown to effectively address security through the dissertation work of its founder, Young and Leveson [8]. STPA-Sec has demonstrated promise for facilitating early security and resiliency requirements generation with traceability to stakeholder prioritized safety and security needs. STPA-Sec has proven its utility for cybersecurity in the defense industry and the DoD. The DoD has adopted STPA-Sec, molding it into Functional Mission Analysis for Cyber (FMA-C). FMA-C is a version of STPA-Sec owned by the USAF which has been tailored to meet the USAF's mission needs [13]. FMA-C is being taught to hundreds of airmen in an effort to assure critical cyber systems and reduce vulnerabilities. The Air Force Chief Information Officer designated FMA-C as the official analysis methodology for the service's fledgling Mission Defense Teams. While the structure and content of FMA-C is very similar to STPA-Sec, its application has been tailored to as-is Information Technology infrastructure. In practice, USAF Mission Defense Teams apply FMA-C on fielded cyber systems to identify mission critical vulnerabilities. The practical application of FMA-C to IT central systems has scoped its focus to this mission need. STPA-Sec, specifically in the tailored approach presented in this work, enables analysis of a conceptual MWS prior to a design solution. It elevates the analysis to highest level by employing systems engineering through systems theory focused on complex interactions in cyber physical systems.

John Thomas and Nancy Leveson have provided multiple free resources for STAMP and STPA. In addition to providing a compilation of STPA research documents, they provide an excellent instructional resource on implementing STPA in their 2013 STPA primer [9], a consolidated resource for implementing STPA. However, while it references a chapter for STPA-Sec, it is incomplete, a gap this work seeks to fill. In March of 2018, after the completion of the research discussed in this work, Leveson and Thomas released a new STPA Handbook [14], an extremely valuable resource for a practitioner to reference. While the STPA Handbook is not specifically written for completing STPA-Sec analysis, the handbook conveys the utility of STPA for analyzing many emergent properties to include safety and security of a given SoI. This handbook goes further than the primer and injects considerations for security; however its primary focus is on STPA and safety rather than STPA-Sec.

Of the STPA-Sec analyses completed to date [11], most are either simplified for presentation purposes or limited distribution due to sensitive and proprietary system information. This work seeks to provide an academic but relevant and complete example (includes all phases and steps of STPA-Sec) to promote utility and enable greater understanding for future practitioners.

III. TAILORED STPA-SEC APPROACH

This work details a tailored STPA-Sec approach as performed for a conceptual level case study analysis of a notional next generation refueling military aircraft, titled the KC-X. This case study is executed during the concept life cycle stage prior to a design to demonstrate STPA-Sec's utility for eliciting early security requirements and supporting early design trades. Often, systems engineering rigor, specifically with respect to security, is not applied until a design has been selected in the system development phase; STPA-Sec purposefully begins in the underdeveloped concept life cycle stage to enable the largest possible trade space of potential solutions. This tailored approach of STPA-Sec addresses security in the same manner as the National Institute of Standards and Technology, NIST, 800-160 which defines security as "the freedom from those conditions that can cause loss of assets with unacceptable consequences" [15]. Our tailored STPA-Sec approach embraces this holistic definition of security enabling the system engineer to elicit system considerations beyond just security, as an asset is defined as anything of value. The NIST definition of security and this tailored STPA-Sec approach are concerned with the protection of assets that include, but are not limited to, security as a subset of concerns of business or mission. This case study demonstrates this through eliciting not only security requirements for the KC-X SoI, but safety resiliency and survivability requirements as well.

The data used to conduct this case study is sourced from publicly available USAF next generation tanker, KC-X acquisition documentation [16], [17] and supplemented by the authors as required. Mission specific details have been obfuscated and generalized to allow for widest distribution. The source documentation selected was published before a contractor was selected and prior to any detailed design.

In Fig. 1, we present the STPA-Sec approach as presented in the authors' previous work [7]. STPA-Sec is organized into three levels or phases of systems security requirements analysis: Concept, Architecture, and Design. Each phase (shown in grey) comprises a small number of steps and results in specific security requirements artifacts (shown in green). STPA-Sec begins at the highest level of abstraction with conceptual analysis where the system solution space is widest in which technical, administrative, and/or process solutions are all feasible. The concept phase is focused on eliciting and defining the system's purpose and goal, and produces a list of initial high-level security requirements. The concept phase, while focused on security, also provides a baseline for stakeholder-focused requirements traceability and safety, resiliency, and survivability decision making which is critically important to the U.S. DoD and its MWS [18].

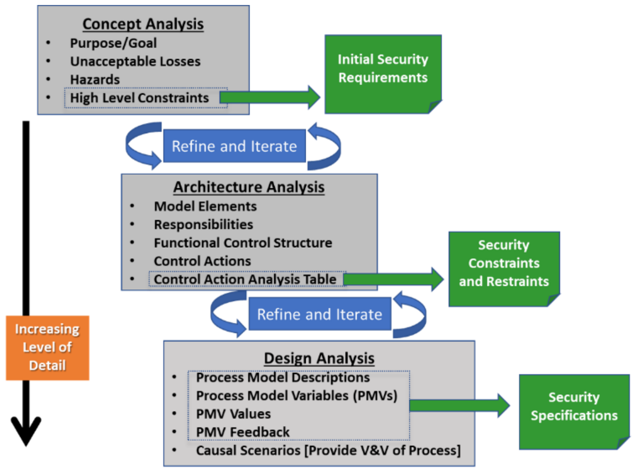
Concept Analysis
Purpose/Goal
Unacceptable Losses
Hazards
High Level Constraints
Initial Security Requirements
Refine and Iterate
Architecture Analysis
Model Elements
Responsibilities
Functional Control Structure
Control Actions
Control Action Analysis Table
Security Constraints and Restraints
Refine and Iterate
Increasing Level of Detail
Design Analysis
Process Model Descriptions
Process Model Variables (PMVs)
PMV Values
PMV Feedback
Causal Scenarios [Provide V&V of Process]
Security Specifications

FIGURE 1. Tailored STPA-Sec overview.

The architecture phase is focused on conducting analysis of the desired SoI at a functional level (i.e. before a form has been selected or implemented). During this phase a list of architectural constraints and restraints is generated from the stakeholder defined initial security requirements. These results enable alternative architectures to be examined and compared for feasibility and protection capability, as well as other classical considerations. The design phase is responsible for generating preliminary security specifications such that potential forms of the system can be studied and compared. This phase results in a set of more detailed security specifications which define acceptable and non-acceptable system states that can be studied, tested, and verified with confidence. Ultimately, these analytical-based results are evidences which directly contribute to claims of the SoI's security trustworthiness.

As shown in Fig. 1, STPA-Sec is an iterative process; it is not intended to be applied as a checklist approach but as tailorable iterations with increasing levels of detail both within and across each phase where refinement of the various phases, steps, and requirements is expected throughout the analysis process. As a STPA-Sec analysis progresses through the phases and steps, learning occurs which forces previous results to be re-visited, corrected, and updated. Moreover, these learning iterations serve to further verify and validate the entire analysis. A STPA-Sec analysis should also be revisited and iterated throughout the system development lifecycle. This case study provides an example during the concept phase of the system before a design has even been proposed. At this point the architecture and design analysis phases of STPA-Sec are purposefully high-level, however when revisited during the development phase of the system life cycle, a revision would further elaborate the security constraints and restraints and provide more detailed security specifications.

While this work executes a STPA-Sec analysis during the concept life cycle stage, the analysis could be executed on a system at any point in its life cycle. STPA-Sec's highest benefit is achieved prior to a design where cost is the lowest to specify requirements. With the increasing cost and complexity of MWS, it has become more common to extend life cycles through modification and upgrade programs that enhance capability. STPA-Sec would prove useful for these programs as it highlights areas to spend limited security dollars by eliciting security requirements that may not have been considered during the initial design. A STPA-Sec analysis is also useful even when funding is not available for modification to the system design as security requirements elicited through analysis are often able to be satisfied through constraints on the Rules of Engagement (ROE's) for system operation.

A detailed explanation of each phase and step of STPA-Sec is presented along with the case study results and recommendations in the following section. While this work and STPA-Sec focus specifically on security, the holistic systems engineering foundation of an STPA-Sec analysis identifies safety, security, resiliency and survivability requirements. This, in fact, is the strength of STPA-Sec as is it is not intended to limit scope to security, but rather provide requirements for the SoI to satisfy multiple domains.

IV. CONCEPT ANALYSIS

As shown in Table 1, the four conceptually-oriented STPA-Sec steps begin with mission-level analysis to identify and prevent the SoI from entering hazardous system states that could lead to unacceptable losses and mission failure. Beginning systems security analysis at the mission-level allows security engineers to more accurately understand the stakeholders' needs and maximizes the engineering trade space (potential system implementations) as system goals are transformed into constraints (i.e., early safety, security, and resiliency requirements).

TABLE 1. STPA-SEC concept analysis.

Step	Description
1. Define the SoI's purpose and goal	Capture the mission statement and key activities of the system: 1) A System to: (What) 2) By Means of: (How) 3) In Order to: (Why)
2. Identify unacceptable losses	Define high-level, intolerable system outcomes to key stakeholders (e.g., loss of life, injury, damage to equipment, reputation, mission, etc.)
3. Identify hazards	Identify system states that when coupled with worst case conditions lead to unacceptable losses
4. Develop system security constraints	Develop mission-informed security constraints that prevent the SoI from entering hazardous states. These constraints are synonymous with early safety, security, and resiliency functional requirements

A. CONCEPT STEP 1: PURPOSE AND GOAL

The first step of STPA-Sec concept analysis defines the SoI's mission in terms of a purpose and goal from the stakeholders' perspective. Stakeholder involvement is extremely important to an accurate mission statement. For a military environment, the mission commander is the ideal person to provide this input with support from their key staff. STPA-Sec's purpose and goal is very important and the exact word choices dictate future analysis steps.

The format of the mission statement is standardized into three parts: 1) Purpose, 2) Method, and 3) Goal, with the KC-X example shown in Table 2. The first phrase "A System to" is meant to capture the primary purpose of the system (i.e., the What) in a few words. For the KC-X the authors distilled several pages of mission information into a short concise statement, "provide worldwide aerial refueling". This statement was paraphrased from the source documentation 'mission statement' to provide a focused and simplified version. For an STPA-Sec analysis it is beneficial to capture the SoI's mission in as few words as possible.

TABLE 2. KC-X purpose and goal.

1. Purpose	"A System to"	Provide worldwide aerial refueling
2. Method	"By Means of"	Flying, Refueling, and Mission Planning
3. Goal	"In order to"	Enable the Air Force Mission to meet Joint Capability Areas via refueling and airlift: Force Enable, Force Extend, Force Multiply

The method portion of the purpose and goal, "by means of", identifies the key activities or processes the system uses to achieve its purpose (i.e., the How). This is a critical and often iterated step as the verbs chosen here become the controlled processes further analyzed in architecture and design analysis. For the KC-X and complex systems in general, narrowing down to a small set of verbs to cover the broad spectrum of key activities is not an easy task. In this case an OV-5 (the Department of Defense Architecture Framework, DoDAF, operational activity model) was available, but its organization did not directly provide mission activity summary tasks. This step highlights that STPA-Sec can leverage previously completed work from Model Based System Engineering, MBSE, activities such as populating DoDAF views like the OV-5 which hierarchically depicts key mission activities. In this case, the OV-5 grouped activities by ground and air, so it required reorganization into functional groupings in an effort to consolidate the 27 functions into 3 high-level activities. STPA-Sec does not specify how many key activities are appropriate, but it would be extremely difficult to conduct the initial analysis with 27 key activities and likely would be very repetitive. While consolidating to

3 activities is not a requirement, abstracting into as few as possible key overall system functions is a best practice for ease of use. Based on a functional grouping of the activities presented in the OV-5 the key activities for the KC-X are: Flying, Refueling, and Mission Planning. These high-level, yet simple and practical functional activities consolidated tasks in the OV-5 such as take-off, navigate en-route, and participate in mission networks into a more useful and pared-down set of key activities. Since mission execution is highly reliant on systems and data from external sources, mission planning should be included as a key activity for this analysis.

Lastly, the "in order to" identifies the goal, or what mission the system contributes to (i.e., the Why). Capturing the mission that the system supports sounds easy but is often difficult in practice as it is difficult to determine the correct level of "mission" the SoI supports. It can be an easy trap to say, 'enables the Air Force mission', or 'supports the warfighter', but these statements lack appropriate specificity. Specificity provides the necessary context to more precisely define terms such as "security" and "resilience" in a way that supports engineering decisions and actions. For the KC-X, a synthesis occurred between: fulfilling the national defense strategy, meeting the quadrennial defense review, achieving the Joint Capability Areas, and prosecuting the USAF's seven warfighting missions [19], [20]. The decision was made to combine the most relevant of the above in a coherent goal focused on the AF mission decomposed into the primary missions enabled through the desired KC-X system. Accurately defining the desired system's purpose and goal can be challenging. Best results are produced with involvement from key stakeholders such as mission owner(s), operators, and users. Moreover, correctly defining the mission (or business case) provides a baseline for prioritizing and performing security tradeoffs within an operationally-focused paradigm.

B. CONCEPT STEP 2: UNACCEPTABLE LOSSES

The second step of STPA-Sec identifies unacceptable losses–intolerable outcomes as defined by mission and system owners (i.e., key stakeholders). Unacceptable losses should be written at the highest level. Possible system losses should identify what is of utmost value to the stakeholders differentiating from what is nice to have or desired. Unacceptable losses can be mission, personnel, or equipment loss. Common unacceptable losses include loss of life and loss of mission essential equipment. However, losses are not just limited to these, loss of reputation or loss of critical data are examples of unacceptable losses that can be addressed through STPA-Sec. Preventing specified losses is the ultimate goal of security. All other subsequent activities share this common goal. Gaining stakeholder consensus on which losses are most important is a crucial first step in framing the security problem. Moreover, it makes little sense for security engineers to expend precious resources to prevent a loss which stakeholders ultimately are not concerned about. Losses can be re-examined throughout the security analysis and grouped together into the highest level of losses. For the KC-X example, the losses identified are:

Unacceptable Losses

L1: Death or Human Injury

L2: Damage to or Loss of Aircraft

L3: Unable to Complete Primary Mission(s)

In the author's review of other STPA work these were common unacceptable losses shared across many other analyses for complex systems [21]. While an STPA or STPA-Sec analysis is not limited to these three losses, these three are applicable to many complex systems beyond aircraft and military applications. For example, L1 is broadly applicable and more generally corresponds to operator loss, L2 is the SoI, and L3 accounts for functional losses. As a note, since this is a military system, in wartime L2 may allow for a certain number of airframes or a certain amount of damage to become acceptable to ensure the mission can be completed in contested airspace.

Given the importance of unacceptable losses to the mission system and stakeholders, these losses provide critical insight and traceability for follow on STPA-Sec steps, resulting in well-defined and justifiable requirements for safety, survivability, and security.

C. CONCEPT STEP 3: HAZARDS

The third step identifies hazards that can contribute to unacceptable losses. STPA defines a hazard as a system state (or set of conditions) that together with a worst-case set of environmental conditions leads to an unacceptable loss [9]. Environmental conditions can be events such as weather but are defined as any condition impacting the SoI that is outside of the system's boundary (i.e., conditions the system has no control over).

The hazard identified should be within the system boundary and not an environmental condition or external actor. For example, a hazard for an aircraft is not a mountain or weather because the designer of the aircraft has no control over the weather or the location of a mountain. Instead the hazard of the aircraft getting too close to the mountain using STPA-Sec is written as violation of altitude/clearance from terrain. The resulting hazard could result from any one of many environmental conditions: weather, turbulence, improper guidance, loss of navigation, cyber attack, etc. For all these conditions, the hazard as written is still valid and system requirements can be selected to mitigate a larger set of potentially unsafe scenarios which can lead to unacceptable losses, thus covering a broad scope of identified and unidentified environmental conditions. An example of a requirement informed by this hazard is a redundant altimeter system informed by multiple sources of altitude. This safety and security requirement would likely prevent the collision with the mountain (unacceptable loss) independent of which environmental condition (or malicious actor) put the aircraft on a collision course. Cyber attack by threats is one source of scenarios, but it is not the only source.

For the KC-X example, four hazards are identified in Table 3 with emphasis on writing them independent of

TABLE 3. KC-X hazards.

	Hazard to Loss Cross Walk Table	L1 Death or Human Injury	L2 Damage to or Loss of Aircraft	L3 Unable to Complete Mission(s)
H1	Flying to Close to Other Aircraft/Out of Position	X	X	X
H2	Violation of Altitude/Clearance from Terrain	X	X	X
H3	Unable to Evade Enemy Threats	X	X	X
H4	Mission Critical Systems not Functional when Required			X

environmental conditions. Hazards 1 and 2 are written to scope broad groups of likely environmental challenges the aircraft may encounter (e.g., weather, improper navigation, turbulence, pilot error) into controllable activities that combined with these worst-case environmental conditions would likely result in a loss. Hazard 3 is included to capture the hazardous potential of adversaries in an effort to stress the importance of tradeoffs for system survivability. Hazard 4 was included to emphasize the importance of resiliency and security for those systems deemed mission critical (i.e., the flight management system or refueling control subsystem).

As a general rule, hazards should be abstracted up to the highest level possible, and in most cases the list of hazards should be fewer than ten [9]. In practice it is often easier to collect a larger list of hazards and then combine or group similar hazards. Identifying hazards also serves to refine and clarify the list of unacceptable losses, as each hazard should be mapped to one or more unacceptable losses. If a hazard is not mapped to an unacceptable loss, then it is either not a hazard or the list of unacceptable losses is incomplete [9].

Controlling hazards is STPA-Sec's conceptual mechanism for delivering system security based on Leveson's STAMP safety model that associates high-level unacceptable losses as control deficiencies across the SoI (rather than individual component failures) [12]. These control deficiencies manifest themselves as problems between components (interactions) rather than simple mechanical failures. The latter have been the traditional cause of failures in mechanical systems, but studying the former provides much more utility when developing contemporary, large, software-intensive systems where hazardous states are a necessary precondition to loss [14]. For example, if the unacceptable loss is two aircraft colliding, then the associated hazard could be generalized as failing to maintain safe separation between aircraft. If safe separation distances are maintained, the two aircraft should never enter a hazardous state or experience this loss. The need for safe separation is identified through STPA-Sec. Ensuring safe separation is maintained (a control function) throughout operation is a systems problem and can be addressed through systems engineering and secure systems engineering. Note, there are several different ways (scenarios) the violation might occur beyond operator error. The air traffic control system might be hacked or attacked, or one or both of the aircraft might be subjected to a cyber-attack. Likewise, there are any number of mitigations that can be used to address the hazard. At this conceptual stage of the system engineering process, the goal is merely to identify the hazard (safe separation not enforced). In other words, loss prevention functionality (safe separation enforcement) is now identified and the remainder of the engineering process can focus on developing a suitable solution to enforce the required safety and security functionality. This approach allows engineers to handle safety and security in the same manner that all other system properties are addressed. STPA-Sec highlights complex, highly interactive scenarios involving operations, processes, and operators rather than just focusing on components as these other factors are often neglected in traditional analysis methods yet are contributory to many losses.

D. CONCEPT STEP 4: CONSTRAINTS

The fourth step of the concept analysis phase is developing system security constraints that prevent the SoI from entering one of the previously identified hazardous states. These constraints are restrictions placed on the system (and implemented via the security architecture) to bound operation within acceptable parameters defined in phase two and three. These constraints are the output of the concept analysis phase which inform early security requirements that are directly traceable to key mission needs for the SoI. The constraints identified for the KC-X example are shown in Table 4. In the same way hazards are mapped to losses, each constraint should be mapped to one or more hazard and each hazard should be mapped to at least one constraint. Throughout our analysis we found most constraints mapped to a single hazard; however, some hazards had multiple constraints. If a constraint cannot be created for a given hazard, it is likely that the hazard is outside the system boundary and may be an environmental condition.

TABLE 4. KC-X constraints.

	Constraint	**Mapped Hazard**
1	Aircraft must maintain minimum safe separation distance	H1
2	Must have minimum mission critical safety systems functional to attempt AR	H1
3	Aircraft must maintain minimum safe altitude limits	H2
4	Must have minimum mission critical safety systems functional for terrain flight	H2
5	Must maintain integrity of mission critical warning and deterrence systems	H3
6	Mission critical systems must be available when required to perform primary mission	H4

In general, constraints can often be thought of as simply re-writing hazards in the form of positive restrictions. These restrictions should prevent the system from entering previously identified hazardous conditions (i.e., an unsafe or insecure system states). For example, KC-X constraints 1, 3, and 6 are written in this relatively straight forward manner to specifically address hazards H1, H2, and H4, respectively. This approach also highlights the need for critical mission systems to operate as expected as described in KC-X constraints 2 and 4. Lastly, it is also important to note that defining systems security requirements in this manner often results in stakeholder-driven safety, security, and resiliency requirements which are reasonably well justified and ensure the SoI's mission-essential capability.

E. CONCEPT OUTPUT: INITIAL SECURITY REQUIREMENTS

The first four steps of STPA-Sec begin to specify acceptable and non-acceptable system states which can eventually be formally tested and verified when the architecture is developed. At the completion of this concept phase the list of constraints is high-level and not necessarily security specific, applying more broadly to safety, security, and resiliency. As the STPA-Sec analysis is continued into the next phases these can and will be refined to further specify measurable and verifiable requirements for safe and secure system operation. The outputs of STPA-Sec can be used to inform and refine early MBSE efforts, with some future work planned to provide executable outputs of STPA-Sec [11]. These refined safety and security focused constraints provide the baseline for measurable safety, security, and resiliency requirements which are also important for system survivability [18].

V. ARCHITECTURE ANALYSIS

STPA-Sec architecture analysis is a continuation of the concept phase and examines the SoI at the functional level rather than a form-specific implementation as is often the case in cybersecurity analyses. Approaching the analysis from a functional rather than physical implementation maintains the largest trade space for potential solutions and helps ensure the desired system functionality can be implemented without unnecessary architectural and design constrictions.

Table 5 details the necessary steps to perform STPA-Sec architecture analysis. The majority of this analysis involves the creation of a Functional Control Structure (FCS) demonstrated in Fig. 1. The FCS can be created at various levels of abstraction such that an entire SoI is represented as a single model (e.g. Fig. 2), or the FCS can be decomposed into multiple sub-models to more specifically understand the execution of the SoI's key activities (e.g. Figs. 4-6). As a reminder, the key activities were defined in the concept analysis phase step 1 as the verbs composing the method, or 'By Means of' section of the SoI purpose and goal. Referring to Fig. 2, STPA uses functional decomposition to thoroughly understand critical relationships between actors and processes represented as Control Actions (CA) in feedback loops. After producing an FCS, STPA-Sec enumerates all required CAs followed

TABLE 5. STPA-SEC architecture analysis.

Step	Description
1. Identify Model Elements	Identify actor(s), controller(s), and controlled process(es) for the SoI at the desired level of abstraction.
2. Identify Each Model Elements Responsibilities	Capture the description and actions planned to be taken for the model elements identified.
3. Draw the FCS	Provide a visual functional-level depiction of the SoI. Depicts the model elements and control relationships between them.
4. Identify Control Actions (CA)	Captures (in verb form) the actions necessary for each element to execute its responsibilities.
5. Complete the Control Action Analysis Table	The CA table systematically enumerates which hazards are caused by each CA identified in step 4.

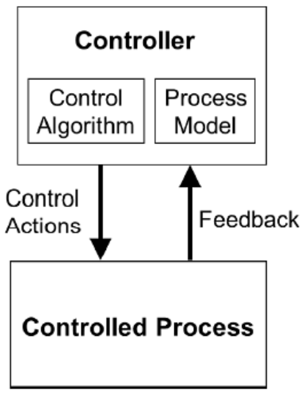

FIGURE 2. STPA fucntional control structure example. from [9].

by an analysis of their criticality and specifically how they contribute to preventing the SoI from entering hazardous states. The output of STPA-Sec architecture analysis identifies potentially hazardous or unsecure CAs for a system architecture. These CAs help further refine system level security requirements within a system architecture.

A. ARCHITECTURE STEP 1: MODEL ELEMENTS

The first step of this phase begins populating the FCS with model elements. The FCS model elements include actors, controllers, actuators, sensors, and controlled processes. A standard format of an FCS is shown in Fig. 2 with the KC-X shown in Fig. 3. Starting from the bottom of the model, the controlled process(es) is(are) the previously identified (Concept Analysis step 1) activities the system uses to achieve its purpose. Beginning with these "How" verb activities, the primary task in step 1 of STPA-Sec architecture analysis is identifying the controller and actors responsible for performing the process. Actors are the operators managing the process and providing inputs to the system. The controller is system specific, but often in high-level FCS models is merely represented as a computer. For the KC-X example the actors are aircrew and the controller is the flight

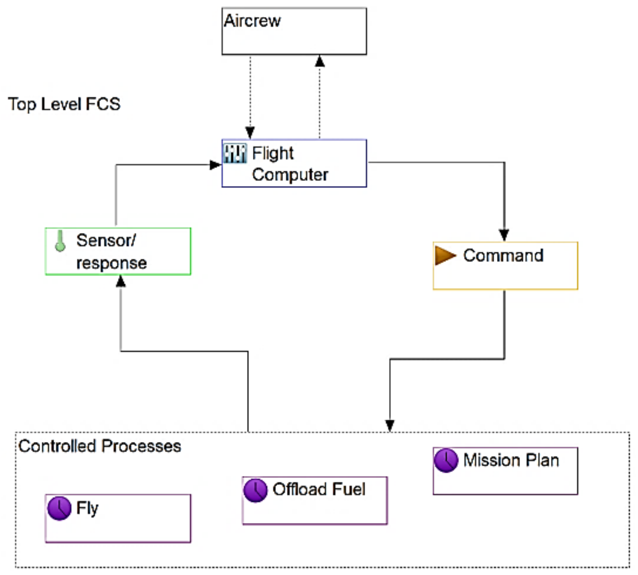

FIGURE 3. **KC-X top level functional control structure.**

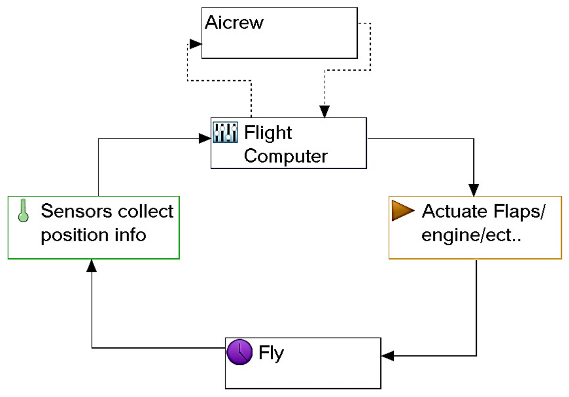

FIGURE 4. **KC-X key actitivy: Fly FCS.**

computer as shown in Fig. 3. Note that an FCS model is not required for identifying model elements, but it is often helpful to sketch an initial FCS to ensure the model elements and their relationships are properly accounted for.

B. ARCHITECTURE STEP 2: RESPONSIBILITIES

Once model elements are identified, then responsibilities for the model elements are populated in step 2. This step captures the actions required of each element for executing the activity (i.e., the controlled processes). These responsibilities can be identified from operational or system documentation, and/or from discussions with users, system SME's, and other stakeholders. Once a list of responsibilities is populated, they are assigned to the appropriate actor for each controlled process (or action).

This step proved to be particularly challenging for the KC-X example due to the analysis being completed at the conceptual level. For example, it was difficult to define the responsibilities without restricting the results to a physical form implementation. In other words, responsibilities are often generated based on what an existing actor or controller did in the previous system where most responsibilities listed are merely duty descriptions of crew members based on previous tanker aircraft operations. Utilization of these crew member specific responsibilities restricts the trade space of the future system. For the KC-X, the documentation listed a boom operator as responsible for accomplishing refueling and more specifically providing alignment cues for the receiver aircraft. While a valid responsibility, it assumes the KC-X solution relies on a human for boom operation and navigation cues. For a next generation platform, it is just as likely that the task of navigation cues could be automated and performed by a computer system. In a conceptually oriented STPA-Sec, practitioners should exercise caution when assigning responsibilities such that the solution space is not unnecessarily scoped down to thinking only how the task is currently executed. Each responsibility should become a line on the FCS in step 3 as each line represents a necessary relationship for the exchange of information for both CAs and feedback. In step 4, these relationships officially become control actions. This sequence is precisely why it is important to capture the responsibilities in step 2 and why identifying CAs in step 4 should cause the STPA practitioner to revisit the responsibilities and FCS from steps 2 and 3.

C. ARCHITECTURE STEP 3: DRAW THE FUNCTIONAL CONTROL STRUCTURE (FCS)

Step 3 organizes the previously identified model elements into a FCS (i.e., a model) by formalizing control relationships. Step 1 already populated the model elements (boxes) for the FCS. Control relationships (and the FCS as a whole) depict who/what is issuing commands (the controller), who/what is executing the commands (the actuator), and who/what is providing feedback (the sensor). Fig. 3 illustrates these basic elements for the KC-X as previously described in step 1 and shows the relationships between them at the highest level. While depicting this level of analysis in an FCS can seem trivial, Fig. 5 depicts the KC-X mission

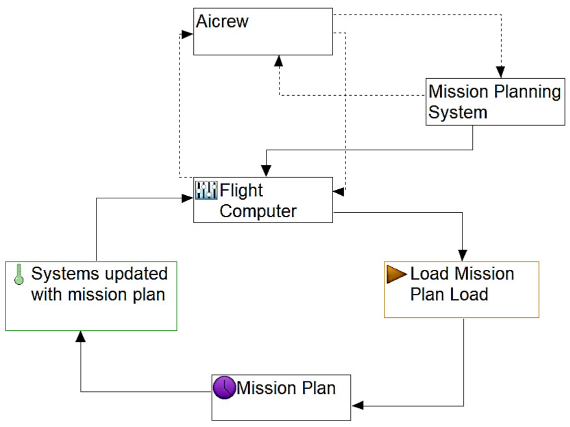

FIGURE 5. **KC-X key actitivy: mission plan FCS.**

planning's more complex relationships, where for example, multiple actors can issue commands to the controller, thus additional relationships need to be modeled. In the KC-X mission planning activity, both the pilot and an external mission planning software have direct inputs to the flight computer. The flight computer may auto-read a mission plan from a cartridge or the pilot may manually enter components of the plan. Fig. 6 depicts the FCS for the offload fuel key activity. When increasing the level of detail, the FCS will become much more complex; however, the primary objective of creating a FCS is to manage the complexity of the system by accurately depicting key elements and control relationships. Thus, determining the appropriate level of detail for the FCS is a tradeoff.

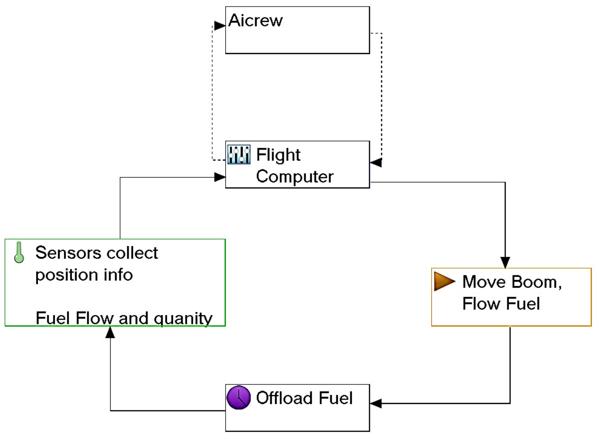

FIGURE 6. KC-X key actitivy: offload fuel FCS.

D. ARCHITECTURE STEP 4: CONTROL ACTIONS (CA)

Step 4 identifies CAs for the system. A CA is a terse verb (action) statement capturing the execution of a function (or task) necessary to control the subject process. This step, in conjunction with the CA analysis performed in step 5, are the most important (and challenging) actions performed in this phase. Populating the list of CAs begins with decomposing the responsibilities previously identified in step 2 into terse action statements that an actor or computer performs to manage the controlled process. The primary challenge is determining what level of abstraction is appropriate for the CAs, because CAs exist at many levels of abstraction. The level of abstraction should be dictated by the overall level of detail for the analysis. CAs should be identified for each activity, and thus for each FCS. The CA's identified for the KC-X example are shown in Table 6. Of note, the list of CAs for the KC-X was originally much longer but the authors purposefully (and painstakingly) abstracted up to the highest level. Tools currently under development by a variety of individuals and organizations should greatly ease this task in the future [11].

Specifically, for the *fly* key activity, multiple CAs were considered, quickly highlighting the challenge of determining the correct level of abstraction. CA1 and CA2 for example, started from a much larger list of many activities required to fly (in order of decreasing level of abstraction): navigate route, change heading, increase bank angle, maneuver yoke, increase speed, increase throttle position setting. The list of potentially valid CAs for the activity *fly* is extensive. After brainstorming this list, similar activities were aggregated as much as possible and the three activities of Position Maintenance, Velocity Maintenance, and Communicate were chosen to capture the much larger subset of potential CAs. Note, for an analysis of a preliminary system design it may be more appropriate to capture the CAs at a lower level of abstraction and thus a much longer list. The mission plan FCS had many potential tasks under consideration for the CAs as well. CAs 7 and 8 demonstrate the link between CAs and responsibilities. Responsibilities are listed in the description section of the list. While not exhaustive of all potential tasks and responsibilities, it provides a good description of efforts executed in the Mission Plan FCS. For this FCS the authors chose to aggregate the list of potential actions into two CAs: Prepare OPS and Distribute OPS.

Of significance to a conceptual STPA-Sec is the performer for the CAs. In the KC-X example, specific and considerable effort was made to remain as solution-agnostic as possible. KC-X CAs 4-6 provide an example: in aerial refueling, operators are familiar with precontact, contact, and breakaway commands as issued by the boom operator. Since this analysis is for a future solution, specifying aircrew as the performer limits the potential solution space where it is feasible for a future computer-controlled system to provide some of these CAs. Or the future KC-X may implement a hybrid solution with both humans and computers. Thus, as much as possible, efforts should be made to not restrict the trade space of potential solutions with prior operation biases when conducting a conceptual analysis.

It can be challenging to understand completeness for this step, but a good indicator of a sufficient CA list is when all the activities for a given FCS can be completed with the listed CAs. This becomes evident when executing causal scenarios (discussed in the design analysis phase step 5). If the scenario can be executed on the given FCS with the identified CAs then it's likely the CAs listed are sufficient. This again demonstrates that STPA-Sec is truly nonlinear, and therefore must be performed iteratively.

E. ARCHITECTURE STEP 5: CONTROL ACTION ANALYSIS TABLE

The fifth step of STPA-Sec architecture analysis is populating a CA analysis table, Table 7. This step requires a thorough analysis of each CA identified in Table 6 (step 4) and enumerates what, if any, violations of the CAs can cause the system to enter a hazardous state.

Each CA is evaluated across four conditions. The first condition asks what happens if the CA is not provided. In most cases this violation contributes to the system entering a hazardous state as most CAs are designed to be executed

TABLE 6. KC-X control actions.

Control Action	Activity	Performer	Description
1. Position Mx	Fly	Aircrew/ Computer	Adjust position- heading change, takeoff, land, climb, descend. Computer included for autopilot functions.
2. Velocity Mx	Fly	Aircrew/ Computer	Change velocity- accelerate, decelerate, climb, descend. Computer included for autopilot functions.
3. Communicate	Fly	Aircrew/ Computer	Radio and digital (i.e. ACARS, IFF) to other aircraft, ATC and ground assets. Access and communicate in net-centric environment.
4. Precontact	Offload Fuel	Aircrew/ Computer	Instructing both crews on proper position to begin aerial refueling. Solution independent to allow for human direction or computer aided position information.
5. Contact	Offload Fuel	Aircrew/ Computer	Receiver connected to begin refueling. Solution independent of human vs. computer to allow automation as desired.
6. Breakaway	Offload Fuel	Aircrew/ Computer	Command to disengage either when complete or in case of emergency. Solution independent of human vs. computer to allow automation as desired.
7. Prepare OPS	Mission Plan	Aircrew/ external mission planning system	Reviews mission tasking, intel, and weather. Interacts with external mission planning system to create mission plan file.
8. Distribute OPS	Mission Plan	Aircrew/ Computer	Aircrew inserts cartridge into jet, also provides crew briefings and coordination for mission plan. Computer distributes mission plan files to aircraft systems.

for safe, secure, and efficient system operation. For the KC-X, all but CA 5 have the potential to contribute to hazardous states when the CA is not provided when required. CA 5's exception is discussed in the following paragraph. CA 1 and 2 provide an easy example of not providing a CA violation contributing to the system entering a hazardous state. If position maintenance and velocity maintenance (the two CAs required to execute the flying functions of the aircraft) are not provided and the aircraft is in a critical phase of flight where pilot or computer input is required, a hazardous system state will result. Since an aircraft cannot execute its mission without performing the flying function, it is not surprising that not providing CAs 1 and 2 could result in H1, H2, and H3.

The second condition asks what hazardous system states can occur if the CA is provided under the wrong context. At first this condition may sound counterintuitive, as one may question why CAs can cause the system to enter a hazardous state when performed. KC-X CA 5, Contact, provides an example of a potential hazard when the CA is provided, under the wrong conditions. When combined with unsafe environmental conditions (specifically the receiver out of position), it can be hazardous to execute the contact CA. If the CA is issued when the refueler or receiver aircraft is out of position, H1 and a resulting collision could occur, resulting in a loss. Of course, the pilot might catch the error and intervene. However, before dismissing the condition and assuming the pilot will catch and correct this error, it is important to consider that the pilot could also be distracted or otherwise unable to intervene (e.g. suffering spatial disorientation). The engineering goal is to find a design that removes the opportunity for this CA to be provided in all but the intended context. However, if a suitable solution cannot be identified, then engineers must place an additional requirement for a clear and unambiguous warning that the CA was issued under incorrect context.

The third condition analyzes the result of a CA provided too late, too early, or out of sequence. This condition is relevant for specific activities where a violation of that timing or sequence can cause a hazardous system state. For the CA 6, Breakaway, provides an example of this. The Breakaway Command is issued during refueling when the aircraft enters an unsafe position. If the breakaway CA is provided too late, a hazardous system state of flying too close or out of position (H1) is likely to occur and lead to an unacceptable loss.

The fourth condition analyzes the results if the CA is stopped too soon or applied too long. KC-X CA 3 provides an illustration of a hazard for this scenario. If communication is stopped too soon (or incomplete) a hazardous system state could result. For example, if radio instructions are provided from the refueler to the receiver to descend and decrease speed, but communication is clipped before the decrease speed command is provided, then when the maneuver begins the H1 hazard is likely to occur if only one aircraft decreases speed.

F. ARCHITECTURE OUTPUT: SECURITY CONSTRAINTS AND RESTRAINTS

The results of STPA-Sec architecture analysis (specifically the CA analysis table) are output as security restraints and constraints shown in Table 8. The CA analysis table captures a multitude of potentially hazardous states to determine design criteria and increase the robustness of the system against these failure modes. The design criteria can then be translated into constraints and restraints that further refine the early security requirements from the concept phase. This data informs an initial ''design-to'' criteria which is further developed during STPA-Sec design analysis. Table 8 shows an example of this

TABLE 7. KC-X control action analysis table.

CA#	Control Action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order	Stopping too soon/applying too long
EXAMPLE		Not Providing CA-1 is hazardous if (CONDITIONS) [(hazards associated) H1, ect]	Providing CA-1 is hazardous if (CONDITIONS) [(hazards associated) H1, ect]	Providing CA-1 is hazardous if too early/too late, or wrong order in (CONDITIONS) [(hazards associated) H1, ect]	Providing CA-1 is hazardous if stopped too soon, or applied too long in (CONDITIONS) [(hazards associated) H1, ect]
1	Position Mx (Aircrew)	Not Providing Position MX is hazardous if in a critical phase of flight [H1, H2, H3]		Position MX is hazardous if done too early or too late in a critical phase of flight [H1, H2, H3]	Position MX is hazardous if stopped to soon or applied to long in a critical phase of flight [H1, H2, H3]
2	Velocity Mx	Not Providing Velocity MX is hazardous if in a critical phase of flight [H1, H2, H3]		Velocity MX is hazardous if done too early or too late in a critical phase of flight [H1, H2, H3]	Velocity MX is hazardous if stopped to soon or applied to long in a critical phase of flight [H1, H2, H3]
3	ATC & Intra-flight Communicatio n	Not Providing Communication is hazardous if in a critical phase of flight (takeoff, landing, joining refueler) [H1, H3]		Communication too late is hazardous if in a critical phase of flight (takeoff, landing, joining refueler) [H1, H3]	Communication stopped too soon (clipped transmission) is hazardous if in a critical phase of flight [H1, H3]
4	Precontact	Not providing Precontact is Hazardous as an aircraft could be out of position and damage equipment [H1,H4]		The wrong sequence for Precontact is hazardous if in a critical phase of refueling setup [H1,H4]	
5	Contact		Providing Contact is Hazardous if attempted during an unsafe position [H1]	Providing Contact out of sequence is Hazardous if attempted during an unsafe position [H1]	
6	Breakaway	Not providing Breakaway is Hazardous if unsafe position occurs [H1]		Not providing Breakaway on time is Hazardous if unsafe position occurs [H1]	
7	Prepare OPS	Not providing Prepare OPS is Hazardous in almost all scenarios (no planned route, no deconflicts, no mission plan loaded on systems…) [H1,H2,H3,H4]			
8	Distribute OPS	Not providing Distribute OPS is hazardous in almost all scenarios (no filed flight plan, no crew briefing, no mission plan loaded on systems…) [H1,H2,H3,H4]	Providing Distribute OPS is hazardous when malware or intentionally incorrect information is distributed to systems [H1,H2,H3,H4]		

but for the sake of brevity does not provide an exhaustive list of all potential security constraints.

The downside to the high-level CA's identified in step 4 is that they lend themselves to less specific results in the CA analysis table. High-level CAs produce very similar scenario impacts that would likely be more diverse with a lower level set of CAs. The results of this effort accomplished on a lower level set of CAs may provide more actionable insights to inform more specific security constraints and restraints. For example, the CA table for Fig. 3 high-level FCS would be significantly different than the CA table for FCS Figures 4, 5, or 6. However, conclusions are still available from this higher level CA analysis. For the KC-X example, the results of the CA analysis table illuminate the importance of the security and reliability of *fly* CAs 1-3, as 3 out of 4 scenario violations result in hazardous systems states corresponding to hazards H1-H3, and the associated unacceptable losses may ensue if these CAs are omitted or executed improperly.

VI. DESIGN ANALYSIS

The STPA-Sec design analysis phase studies the specifics of a control action using relatively simple process models and scenarios. These process models describe the decision

TABLE 8. Example of security constraints and restraints.

Hazardous Control Action	Required System Constraint
Not Providing POSITION MX Commands is hazardous if in a critical phase of flight [H1, H2, H3]	POSITION MX commands must be provided during critical phases of flight
POSITION MX commands are hazardous if done too early or too late in a critical phase of flight [H1, H2, H3]	POSITION MX commands must be executed within a specified time of the maneuver requirement
Providing CONTACT is hazardous if attempted during an unsafe position [H1]	CONTACT command must only be provided if both aircraft are in a safe position ready for AR
Providing CONTACT out of sequence is hazardous if attempted during an unsafe position [H1]	CONTACT command must not be issued or received after the BREAKAWAY command has been issued until the aircraft have resumed a safe position

logic, key variables, and acceptable variable values associated with each CA in a systematic and straight forward fashion. Additionally, this analysis identifies which feedback mechanism is responsible for providing the process model variable values (e.g., a sensor or computing mechanism). STPA-Sec design analysis enables a more thorough understanding and specification for CAs which prevent the SoI from entering potentially hazardous and unsecure states. The steps of design analysis are captured in Table 9.

TABLE 9. STPA-SEC design analysis.

Step	Description
1. Develop Process Model Descriptions	Describes the decision logic "in plain English" for executing a given CA.
2. Identify Process Model Variables (PMV)	Process Model Variables are measurable indicators of the conditions that trigger a CA.
3. Identify PMV Values	PMV values are all the possible values a PMV can be assigned both acceptable and hazardous.
4. Identify PMV Feedback	Identifies which sensors provide PMV values to the actors and controller for decision making.
5. Carry out Causal Scenarios	Brainstorm how a specific implementation of the system may be compromised. Identifies critical CAs and validates the thoroughness of the model, CAs, and constraints.

The focus of this phase demonstrates the utility of STPA-Sec to a notional warfighting system through a case study example early in the system lifecycle. When conducting a conceptual analysis as for the KC-X, the high-level of abstraction chosen for CAs does not lend itself to a fully elaborated design analysis; thus, this section describes and illustrates the steps of the design analysis phase but does not include a detailed analysis of every high-level control action.

A. DESIGN STEP 1: PROCESS MODEL

Each process model should briefly describe the scenario of interest and focus on when to execute a given CA. Process models identify specific system elements and potential responses to CAs along with any assumptions about the controlled process. Step 1 of STPA-Sec design analysis develops process model descriptions shown in Table 10.

TABLE 10. KC-X process model descriptions.

Control Action	Key Activity	Process Model Description / Decision Logic
1. Position Mx	Fly	Execute Position Mx during critical phases of flight
2. Velocity Mx	Fly	Execute Velocity Mx during critical phases of flight
6. Breakaway	Refuel	Issue Breakaway when unsafe position

Process model descriptions capture the decision logic that defines how and when controllers execute CAs.

During this step, it is advantageous to first generate process model descriptions for CAs determined as potentially hazardous from the completed STPA-Sec architecture analysis (step 5, CA analysis table). It is important to prioritize the identification and definition of process model descriptions because a complex system may have a large (and potentially overwhelming) number of process model descriptions. In accordance with this recommendation, for the KC-X example an abbreviated set of CAs was chosen for design analysis to illustrate the steps.

B. DESIGN STEP 2-4: PROCESS MODEL VARIABLES, VALUES, AND FEEDBACK

In step 2, Process Model Variables (PMVs) are described as data which indicate system states. For the KC-X example the Breakaway CA is chosen to illustrate the steps of STPA-Sec design analysis as shown in Table 11. Since Breakaway is issued when an unsafe position occurs between the refueler and receiver, an appropriate process model variable is separation distance.

Once an architecture has been selected and a detailed design is being considered, step 3 can provide more specific PMV values (e.g., 10 ft.). It is critical these values are properly understood to specify potentially hazardous systems states. For the high-level KC-X example with separation distance as the PMV, we sought to establish values that were simple yet enclosed the entire range of values since we do not have a specific solution yet. Rather than choosing a specific range, we chose constrained values that informed the

TABLE 11. KC-X full process model description.

CA	Process Model Description	Process Model Variables	Process Model Variable Values	Feedback Information
Breakaway	Issue Breakaway when unsafe position	Separation Distance	In bounds, out of bounds, unknown	Altimeter warning, proximity warning, eyeball

action: in bounds, out of bounds, and unknown. For the KC-X Breakaway CA, this set of PMV values is more useful than a specific range because we do not have a specific solution yet. These allow for simpler discrete logic commands to be developed (i.e., in bounds - do not issue CA, out of bounds - issue CA, unknown - issue CA).

Step 4 identifies the sensors which are responsible for generating the PMV values (i.e., data) to include conventional sensors, personnel, computer systems, etc. For the KC-X the sensors providing this feedback are an altimeter warning system, proximity warning system, and the aircrew visual cues. This step lends insight into which feedback sensors are critical to monitor for potentially hazardous states and enforce CAs for cybersecurity and assurance.

STPA-Sec design steps 1-4, process models, and PMVs help specify functional logic informing subsystem and component implementation as preliminary design specifications that need to be secured and assured.

C. DESIGN STEP 5: CAUSAL SCENARIOS

Step 5 of STPA-Sec design analysis is the generation of causal scenarios to further study the impact of environmental conditions (previously explored during concept analysis). More specifically, the causal scenarios are used to understand and assess how the constraints might be violated through single point failures or human mistakes. For example, why might a breakaway command not be issued when aircraft proximity violates minimums? One scenario might be that the warning mechanism was compromised through cyber attack. Of course, the solution would be to design a more robust warning mechanism. However, a more subtle deficiency might be the inclusion of logic that switched on the breakaway mechanism only when the aircraft entered air refueling mode as determined by aircraft expected airspeed limits. Under emergency circumstances, the aircrew might need to attempt a refueling outside of published air refueling limits, but within aircraft performance limits. Akin to tabletop red teaming, causal scenario generation is typically conducted by system experts, well-qualified users, and threat analysts with the goal of identifying plausible scenarios (or conditions combined with effects outside the system boundary) that violate a constraint or restraint. In this step the practitioner can use their desired tools to analyze the CA including but not limited to: attack path analysis, attack trees, failure mode effects and criticality analysis, and/or wargaming. It is important to stress that threat-based scenarios are only one type of scenario. Moreover, one of the advantages of STPA-Sec is that specific threat intel is not required to reason about security properties of the SOI. In fact, until protocols and architectures are specified, threat-based scenarios will likely be limited. For examples and additional instruction see [14].

For the KC-X example, a causal scenario analysis is generated for the breakaway CA. This specific example of the breakaway CA was chosen to illustrate a well-designed process model; often causal scenarios will illustrate the potential for an undesired impact and require rework of previous steps to rectify the issue. Breakaway is an excellent example of a well-designed PMV; regardless of the threat (i.e. an environmental condition) thought up in a causal scenario, the system has a robust design to respond. No matter the external conditions, the breakaway PMV feedback indicates in bounds, out of bounds, or unknown. For all scenarios where the feedback assigns a value of out of bounds or unknown the breakaway CA will be issued. This means that turbulence, improper receiver position, poor refueler maneuvering, engine malfunction, and any other undesirable environmental conditions will not violate this process model as designed. Even for unknown scenarios, the controller (human or computer) will attempt to determine the system state and then if still undetermined, issue the breakaway command.

In a general sense, this final step serves to provide verification and validation for the thoroughness of the STPA-Sec analysis effort. In this way, changes or additional constraints are often identified as part of the causal scenario step when attempting to ‘break’ the SoI.

VII. STPA-SEC UTILITY ASSESSMENT

This section presents a subjective assessment of STPA-Sec’s utility for complex weapon systems. Table 12 provides a summarized assessment for each phase.

The concept analysis phase provides the greatest return on time investment. This phase is easy to execute with very little STPA-Sec knowledge in a small working group of key

TABLE 12. STPA-SEC utility asssessment.

	Systems Security Oriented STPA-Sec Phases		
	Concept Analysis	Architecture Analysis	Design Analysis
Purpose	Determine Security Requirements	Determine Design-To Criteria	Determine Build-To Criteria
Difficulty	Easy	Moderate	Moderate-High
Level of Domain Expertise Req’d	Novice	Advanced	Expert
Level of STPA Expertise Req’d	Low	High	Moderate
Amount of STPA Instructional Materials Available	Numerous	Some	Few
Duration	Hours	Days	Weeks
Number of Steps	4 Steps	5 Steps	5 Steps

stakeholders. Additionally, most STPA instructional material and examples are focused on this phase. Establishing an agreed upon purpose and goal along with unacceptable losses from the key stakeholders is very powerful for shaping system security requirements and enabling traceability. STPA-Sec concept analysis prevents the common pitfall of 'securing the wrong thing'.

STPA-Sec architecture analysis is more challenging and time consuming than concept analysis, especially when the system is decomposed to lower functional levels. The level of abstraction chosen for this analysis highly influences the time required. The FCS and list of CAs as executed for the KC-X example were kept at a high-level, if these were decomposed to lower levels of abstraction, the lists of CAs and the CA analysis table could easily become 5-10 times its current length. While this effort would require more domain expertise, the additional information would inform more detailed design-for criteria while also offering the benefit of traceability to key mission activities and prevention of unacceptable losses. While not as clearly articulated as concept analysis in instructional material, the STPA handbook, primer, and other slideshow tutorials do adequately address how to perform architecture analysis. However, with the exception of fictionalized educational examples, architecture analysis for actual systems is often proprietary and not shared. These steps often are the hardest to execute for the STPA-Sec novice, and the lack of fully detailed real world system examples adds to this difficulty.

STPA-Sec design analysis is the most detailed phase of STPA-Sec and can require the most time if each CA is fully detailed via process models. The practitioner will require a much more robust understanding of the SoI as PMVs, potential PMV values, and the feedback sensors are often more technical than the previous analysis. However, this analysis pays dividends in its ability to provide early design specifications for components. These specifications are not only presented in clear decision logic, but are traceable back to key stakeholder mission goals and security requirements. Very few of the publicly available STPA materials currently available fully detail this phase of execution. While the steps are not as complex to execute as the architecture analysis phase, design analysis requires the greatest domain expertise along with the most amount of time investment for completion. Additionally, the detailed analysis completed in this phase often drives refinements and changes to previous phases specifically through causal scenario exercises.

VIII. CONCLUSION

This work presents a thorough case study of an STPA-Sec analysis for a notional next generation aerial refueling system, KC-X. This work provides a detailed explanation of each step and recommendations for STPA-Sec's execution. This explanation provides a consolidated resource to enable future practitioners to execute STPA-Sec for their SoI.

This work demonstrates a tailored approach to STPA-Sec which is organized into phases and steps for ease of execution. However, it is not intended to form a checklist for security as many security methods are executed in this fashion and do not result in secure systems. STPA-Sec encourages an iterative analysis where steps are expected to drive changes to previous results in an effort to further refine and specify system security requirements.

STPA-Sec demonstrates utility for eliciting, defining, and understanding security and resiliency requirements for advanced CPS. Further research should expand this example to incorporate SME's from a system program office to increase the level of detail and evaluate the specific requirements generated in the architecture and design phases of STPA-Sec. Additionally, alternate examples for other services or system types such as space systems should be analyzed.

IX. 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] R. Saracco. (Jan. 13, 2016). *Guess What Requires 150 Million lines of Code, EIT Digital*. Accessed: Feb. 2017. [Online]. Available: https://www.eitdigital.eu/news-events/blog/article/guess-what-requires-150-million-lines-of-code/

[2] A. Greenberg. (Jul. 21, 2015). *Wired, Wired Magazine*. Accessed: Apr. 25, 2017. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[3] E. Perez. (May 18, 2015). *CNN* Accessed: Apr. 25, 2017. [Online]. Available: http://www.cnn.com/2015/05/17/us/fbi-hacker-flight-computer-systems/

[4] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Secur. Symp.*, 2011, pp. 77–92.

[5] United States Congress. (Nov. 25, 2015). *Nation Defense Authorization Act 2016 Section 1647*. Accessed: Jun. 1, 2017. [Online]. Available: https://www.congress.gov/114/plaws/publ92/PLAW-114publ92.pdf

[6] W. Young. (Mar. 25, 2014). *System-Theoretic Process Analysis for Security (STPA-SEC): Cyber Security/Mission Assurance*. Accessed: Aug. 22, 2018. [Online]. Available: http://psas.scripts.mit.edu/home/wp-content/uploads/2014/03/Young_STAMP_2014_As-delivered.pdf

[7] M. Span, L. Mailloux, M. Grimala, and W. Young. *A System Security Analysis Approach for Requirements Analysis of Complex Cyber-Physical Systems*. Accessed: Aug. 22, 2018. [Online]. Available: https://www.c-mric.com/wp-content/uploads/2018/06/Martin_Span_Cybersecurity2018.pdf

[8] W. Young and N. G. Leveson, "An integrated approach to safety and security based on systems theory," *Commun. ACM*, vol. 57, no. 2, pp. 31–35, 2014.

[9] N. Leveson and J. Thomas. (Sep. 9, 2013). *An STPA Primer*. Accessed: Aug. 12, 2017. [Online]. Available: http://sunnyday.mit.edu/STPA-Primer-v0.pdf

[10] N. Leveson, "A new accident model for engineering safer systems," *Saf. Sci.*, vol. 42, no. 4, pp. 237–270, 2004.

[11] Massachusetts Institute of Technology. (Mar. 27, 2018). *MIT Partnership for a Systems Approach to Safety*. Accessed: Apr. 8, 2018. [Online]. Available: http://psas.scripts.mit.edu/home/stamp-workshop-2018/

[12] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety* . Cambridge, MA, USA: MIT Press, 2011.

[13] Air Force Cyber College. (2015). *Top-Down Purpose-Based Cybersecurity*. Accessed: Jan. 1, 2018. [Online]. Available: https://www.sans.org/summit-archives/file/summit-archive-1492176717.pdf

[14] N. Leveson and J. Thomas. (Mar. 2018). *STPA Handbook*. Accessed: Apr. 1, 2018. [Online]. Available: http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

[15] R. Ross, M. McEvilley, and J. C. Oren, "NIST special publication 800–160: Systems security engineering," vol. 1, Nat. Inst. Standards Technol., Washington, DC, USA, 2016.

[16] U.S. Air Force. (Feb. 10, 2009). *KC-X Statement of Objectivies (SOO)*. Accessed: Sep. 12, 2017. [Online]. Available: https://myclass.dau.mil/bbcswebdav/institution/Courses/Deployed/TST/TST204/Student_Materials/Jan%202017/Student%20CD/1%20References/11%20Sample%20Documents%20%26%20Templates/SOO%20Tanker%20Sec%20J%202009.pdf

[17] U.S. Air Force. (2007). *System Requirements Document (SRD) for the KC-X*. Accessed: Jul. 20, 2017. [Online]. Available: https://www.defenseindustrydaily.com/files/PUB_2007-01_KC-X_SRD-20070125.doc

[18] Defense Acquisition University. (May 23, 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

[19] U.S. Department of Defense. (Jan. 2014). *Quadrennial Defense Review*. Accessed: Dec. 10, 2017. [Online]. Available: https://www.defense.gov/News/Special-Reports/QDR/

[20] U.S. Department of Defense. (Sep. 2007). *Joint Operating Concept*. Accessed: Dec. 12, 2017. [Online]. Available: https://fas.org/irp/doddir/dod/iw-joc.pdf

[21] Massachusetts Institute of Technology. (Jan. 1, 2017). *MIT Partnership for a Systems Approach to Safety*. Accessed: Jul. 12, 2017. [Online]. Available: https://psas.scripts.mit.edu/home/

[22] W. Young and R. Porada. (Mar. 27, 2017). *System-Theoretic Process Analysis for Security (STPA-SEC): Cyber Security and STPA*. Accessed: Jan. 21, 2018. [Online]. Available: http://psas.scripts.mit.edu/home/wp-content/uploads/2017/04/STAMP_2017_STPA_SEC_TUTORIAL_as-presented.pdf

[23] M. Span, L. Mailloux, and M. Grimaila, "Cybersecurity architectural analysis for complex cyber-physical systems," *Cyber Defense Rev.*, vol. 3, no. 2, 2018.

MARTIN (TRAE) SPAN, III, (M'15) received the bachelor's degree in systems engineering from USAFA in 2012 and the M.Sc. degree in systems engineering from the Air Force Institute of Technology. He has served the USAF as a Developmental Test Engineer responsible for planning and executing complex weapon system test and evaluation. He serves as a developmental engineer and holds the Department of Defense Certifications in systems engineering, science and technology management, test and evaluation, and program management. He is currently an Instructor of systems engineering with the United States Air Force Academy (USAFA), Colorado Springs, CO, USA. He is also commissioned as a Captain in the United States Air Force (USAF). His research interests include systems engineering and systems security engineering. He is a member the Tau Beta Pi honor Society.

LOGAN O. MAILLOUX (SM'18) received the B.S. degree in 2002, the M.S. degree in 2008, and the Ph.D. degree in 2015. He has served the USAF as a Cyberspace Operations Expert responsible for planning and executing network defense exercises, documenting and training computer security best practices, performing test and evaluation of enterprise resource planning solutions, and maintaining distributed simulation infrastructure. He is currently an Assistant Professor with the Air Force Institute of Technology, Wright-Patterson AFB, OH, USA. He is also commissioned as a Lieutenant Colonel in the United States Air Force (USAF) and serves as a Computer Developmental Engineer. He is also a Certified Information System Security Professional, a Certified Systems Engineering Professional, and holds the Department of Defense Certifications in cyberspace operations, systems engineering science and technology management, test and evaluation, and program management. His research interests include systems security engineering, quantum key distribution, cyber-physical systems, and complex information systems. He is a member of INCOSE and ITEA Professional Societies, and, HKN and TBP Honor Societies.

ROBERT F. MILLS (SM'08) is currently a Professor of electrical engineering with the Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, USA. His teaching and research areas include systems engineering, digital avionics, cyber warfare, electronic warfare, computer networks, and digital communications systems. He is a member of Eta Kappa Nu and Tau Beta Pi.

WILLIAM (BILL) YOUNG, JR., received the B.Sc. degree in engineering science, four master's degrees, and the Ph.D. degree in engineering systems from the Engineering Systems Division, School of Engineering, Massachusetts Institute of Technology. He commissioned in 1991 after graduating from the United States Air Force Academy. He earned his wings from Specialized Undergraduate Navigator Training. He was a 2006 graduate of the USAF School of Advanced Air and Space Studies. He is currently a Colonel in the USAF and also commands the 53rd Electronic Warfare Group, Eglin AFB. He is also a Distinguished Graduate of the US Air Force Weapons School. He is also an Instructor Electronic Warfare Officer with over 2400 flying hours in the EA-6B and B-52, including 240 combat hours during Operation ENDURING FREEDOM.