Security Engineering – FY17 Systems Aware Cybersecurity

Technical Report SERC-2017-TR-114

December 7 2017

Principal Investigator:

Dr. Barry Horowitz, University of Virginia

Co-Principal Investigators:

Dr. Peter A. Beling, University of Virginia
Cody Fleming, University of Virginia

Research Team:

Stephen Adams, University of Virginia
Bryan Carter, University of Virginia
Krishnamurthy Vemuru, University of Virginia
Carl Elks, Virginia Commonwealth University
Tim Bakker, Virginia Commonwealth University
Kryzsztof Cios, Virginia Commonwealth University
Georgios Bakirtzis, Virginia Commonwealth University
Aidan Collins, Virginia Commonwealth University
Nancy Mead, Carnegie Mellon University
Forrest Shull, Carnegie Mellon University

Sponsor: DASD(SE)

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004 (Task Order 0072, RT 172).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF (TABLES, SEQUENCES)

EXECUTIVE SUMMARY

The 2017 effort in System Aware Cybersecurity extended the RT-156 research to focus on resilience features that sustain operator control of weapon systems and assure the validity of the most critical data elements required for weapon control. The decision support tool research focused on integrating historical threat considerations as well as risk considerations into the planning for defenses. Specifically, research investigated the threat analysis aspects of the integrated risk/threat decision support process and included the development of new threat analysis methods focused on mission-aware security. The principal goal was to create and update decision support tools to help decision-makers understand the relative value of alternative defense measures.

The evaluation efforts regarding algorithms for enhanced automation for decision support, led by UVA, have reached an advanced state. Development has continued on a first prototype of the HW, SW, and operational emulation of the weapon system to be evaluated by use of the decision-support tools. Results suggest that our "War Room" approach yields SysML representations that both (a) capture mission objectives and system behavior while (b) providing a representative surrogate surface for attack tree application.

The team developed both the methodology and associated toolset with the explicit intention of generality and broad applicability. Development is complete on a first prototype of a HW/SW emulation weapon system created for testing the decision-support tools. The system includes emulation of all major components of an actual weapon system while also allowing the exploration of more complex operational scenarios and attack spaces, including system-of-systems operations and attacks. The cost of the prototype HW/SW emulation was well-suited to the overall project budget. For the weapon system emulation, we derived mission-level requirements using a hierarchical modeling approach through the War Room exercise. This work included reconstructing the hierarchical model of the intelligent munitions systems including: requirements, behavior (activity diagrams), and structure, all the while keeping traceability between the lower levels of the hierarchy and the mission requirements

The team made significant progress on developing the architectural decision support tools. The analysis and modeling methodology takes a mission-centric viewpoint, combining inputs from system experts at the design and user levels utilizing Systems-Theoretic Accident Model and Process (STAMP) to identify potentially hazardous states that a system can enter and reason about how transitioning into those states can be prevented. The SysML Parser is a tool that connects general system descriptions with a graph model of the system that can be "virtually attacked" by a cyber analyst using the Cyber Analyst Dashboard tools. The V1 Parser is a MagicDraw plugin that utilizes the OpenAPI to automatically extract Internal Block Diagram (IBD) structures to GraphML. The tool includes a modeling methodology that ensures the SysML blocks have a sufficient set of attributes for performing attack chain queries. Outcomes this year include developing a deeper understanding of open source cyber attack databases (e.g., CAPEC, CWE, CERT, and CVE), as well as defining and develop SysML modeling constructs and a traceability ontology to effectively capture relations between missions and system, components in the

presence of attack patterns. Key accomplishments for this phase include: (1) use of several different NLP/querying techniques to characterize relationships between attack classes in CAPEC, CWE, and CVE; (2) refinement of GraphML meta-model; (3) development of CYBOK (Cyber Model of Knowledge) model to guide what information from the cyber domain needs to be present in the SysML mission-aware model; and (4) development of the Cyber Analyst Dashboard – V1. The dashboard presents an interactive view of both the "System" and the "Attack Space" and allows for several different levels of automation as well as human/analyst interaction.

Each of the tools is published as a binary and/or executable that can either work independently or jointly. The Dashboard can function directly with CYBOK or independently; for example, the analyst can directly query specific entries in CAPEC, CVE, CWE through the dashboard, without using the automated recommender system that underpins CYBOK.

System Aware Cybersecurity

Recent work focused on by the Software Engineering Institute (SEI) has been directed toward building on SEI's previous efforts in 3 threat modeling methods: STRIDE, Security Cards, and Persona-non-Grata (PnG). Current effort centers on (1) merging Security Cards and PnG into a single hybrid threat modeling method (hTTM) and (2) using hTTM and other methods on the emulated weapon system.

On July 25th, 2017, UVA hosted ARDEC technical staff and OSD, along with the VCU and SEI team members, for a review of the decision-support methodologies and tools and the prototype HW/SW emulation of the weapon system. This review stimulated interest in developing sentinel applications for the weapon system and in addressing issue in the propagation of resiliency by updating prior information and current operational profiles through networks of sentinels. The OSD, ARDEC and UVA team members all agreed that the HW/SW model and highly simplified rendition of a weapon system developed by UVA provides a useful mechanism for making progress while the selected weapon system data is not yet released to UVA. In the last few weeks, the team conducted the "kick-off" meeting for RT-191. This vehicle will provide UVA with the capability to test the RT-172 decision support tools on a hypothetical, but realistic, weapon system.

The following sections cover the principal technical activities and findings of the research.

USE CASE DESCRIPTION

As part of preparing for evaluations of the decision support tool research activities, an initial system use case was developed. The use case is intended to support research related to both decision support tool development and rapid prototyping to help identify potential system resilience solutions. The first-iteration use case selection consisted of a hypothetical land mine weapon system connected to a unmanned aerial vehicle-based surveillance system and a high level command and control system. A description of the system is presented below:

- Purpose: Prevent, when and where necessary, via the use of a rapidly deployable land mine system, adversaries from trespassing into geographic areas that are close to strategically sensitive locations.
- Prohibited Area : 5-10 acres of open field space
- Land Mines: About 50 short range mines distributed over the prohibited area
- Operation: Operator remote-control of individual or groups of mines, based upon surveillance of the prohibited area (operator located 250-500 feet away from prohibited area).
- Prohibited Area Surveillance: The operator has binoculars to support visual observation and also is supported by real-time video information. In addition, the operator receives information from multiple sensors (acoustic, seismic and infra-red) that are designed to detect trespassers. These sensors are co-located with the land mines. Furthermore, the operator receives video-based imagery provided by a separately operated UAV that is flying within line-of-sight of the protected area. .
- Land-mine design features: The land mines are designed so that they provide regular situation awareness reports (seconds apart) to the operator. This includes reports on their location (GPS-based), their on-off status, their acceptance of commands, their actual firings, sensor detection reports, etc. Furthermore, their SW can only be modified by electrically disconnecting their computer from the land mine, and removal results in destroying that computer. Designed this way to avoid debugging related SW errors (now providing collateral value related to possible cyber attacks).
- Requirements for Avoiding Errors: Significant concerns about detonating land mines in cases where non-adversarial people, by chance, enter the prohibited area, and also about failing to detonate land mines when an adversary is approaching the strategically-sensitive location via the prohibited area.
- Operator Functions: The operator can cause individual or designated groups of land mines to detonate through commands sent via the weapon system's integrated communication network, designed to permit needed communications between the land mine system operator, the individual land mines, the command center that the operator reports to, the UAV video collection subsystem, and the UAV pilot.
- Operator Control Station: Hand held computer provides operator observation of weapon status, weapon control inputs, sensor reports, and supports required digital situation awareness-related reporting to the command center and the UAV pilot.
- Command Center Controls: The command center digitally provides weapon control information for the operator (determines weapon system on/off periods, designates periods of

higher likelihood of attack, provides forecasts of possible approach direction to the prohibited area, enables operation with/without UAV support, etc). As determined by either the operator or the command center, out of norm situations can be supported through rapid message communications between command center and the operator.

- Forensics: All subsystems collect and store forensic information for required post-mission analysis purposes
- Rapid Deployment Support: All subsystems enable rapid deployment support features, including automated confirmation testing of the integrated system.
- UAV Video Collection/Distribution Subsystem: Piccolo

A description of the hardware/software prototype for use in identifying and evaluating potential Sentinel-based resilience solutions is presented in the attached PowerPoint presentation (Appendix A). Follow-on activities will include using the prototype for deriving and evaluating new Sentinel design patterns. In addition, it is anticipated that the follow-on decision support tool activities will likely call for additions to the use case capability that will support evaluation of analysis needs that have not yet been identified.

METHODOLOGY FOR ELICITING REQUIREMENTS AND GENERATING SYSTEM MODELS

Assessing the security of Cyber-Physical Systems (CPS) has long been handled in the same manner as that of software security: that is to identify and address individual component threats. These threats are often identified by analysts using a variety of threat detection methodologies. However, it has become increasingly common, when dealing with complex coupled systems, that vulnerabilities are only identified during forensic analysis after a security breach[1][2] or even after detrimental effects have already taken place[3]. This issue is particularly concerning in the realm of safety-critical CPS, where such security breaches can put human lives in immediate danger. This is the case due to the intrinsic interaction between software-oriented control and the physical world in CPS, where the lines between the fields of safety and security become blurred, such that it is necessary to consider them as a single entity when trying to ensure the successful and safe operation of CPS.

An important metric that has often been neglected in the security of CPS is the specific mission, i.e., expected service, that it is intended to perform). Traditional analysis methods are mission-agnostic, vulnerabilities are viewed in the context of whether or not security is breached, regardless of the magnitude of the breach's effect on its mission requirements or possible unacceptable outcome later on. By creating a mission-aware analysis, the steps towards

[1] "Equifax Hackers Stole 200k Credit Card Accounts in One Fell Swoop ...." 14 Sep. 2017, https://krebsonsecurity.com/2017/09/equifax-hackers-stole-200k-credit-card-accounts-in-one-fell-swoop/. Accessed 4 Dec. 2017.

[2] "ics-alert-14-176-02a - ICS-CERT - US-CERT." 27 Jun. 2014, https://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-176-02A. Accessed 4 Dec. 2017.

[3] "USB Malware Targeting Siemens Control Software (Update C) | ICS ...." 2 Aug. 2010, https://ics-cert.us-cert.gov/advisories/ICSA-10-201-01C. Accessed 4 Dec. 2017.

mitigating a vulnerability are taken in a manner that prioritizes the outcome of the mission. This not only includes traditional security solutions, but also resiliency solutions. Resiliency in the context of CPS refers to the ability of the system to continue to provide the expected service despite cyber attacks or other disturbances. This means that on the one end, a vulnerability may be ignored if it has no effect on mission outcome while on the other end, classes of vulnerabilities that could potentially disrupt the mission of the CPS might require extensive preemption and mitigation strategies.

This approach to CPS cyber-security is born out of the need to assure the successful mission of military systems such as Unmanned Aerial Vehicle (UAV) based surveillance, smart munitions, and other platforms with varying capabilities. For example, a small, hand-launched UAV used for tactical reconnaissance likely has significantly fewer threats to mission success than that of a large, strategic reconnaissance UAV used against a nation-state. This concept extends to non-military systems as well. Autonomous vehicles have major security concerns, but the security needs may differ based on the mission it is assigned to perform. In civilian applications, an autonomous highway vehicle might have far greater potential for harming others than an autonomous farming vehicle collecting produce; thus the measures taken to secure each vehicle should differ accordingly. This strategy allows for informed security decisions, especially in resource-limited scenarios, and prevents securing a system from becoming unmanageable.

With the concepts mentioned above in mind, we have been proposing a new, top-down analysis and modeling methodology that takes a mission-centric viewpoint to safety and security of CPS. This methodology combines inputs from system experts at the design and user levels utilizing Systems-Theoretic Accident Model and Process (STAMP) [Leveson, N. (2011)] to identify potentially hazardous states that a CPS can enter and reason about how transitioning into those states can be prevented. By focusing on the intended mission, this methodology can be applied to both existing and yet-to-be designed systems, which allows for security analysis to occur earlier in the design cycle. According to Figure 3.1, this allows for security solutions to have both greater impact on performance and reduced cost of implementation. Additionally, this proactive, data-driven approach is in contrast to the reactive approach employed by other security strategies.

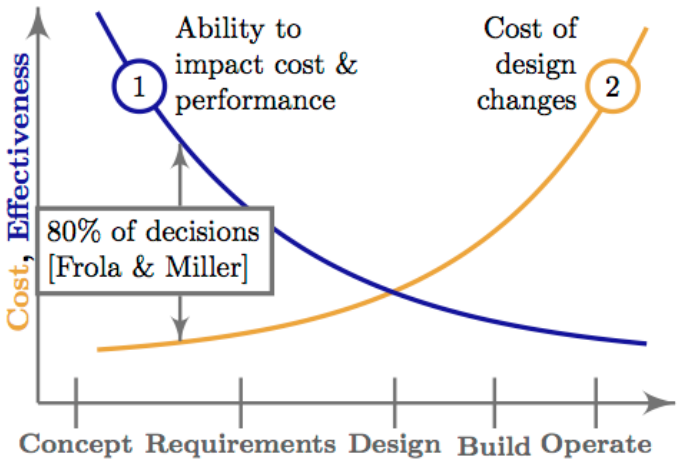


Figure 1 Decision Effectiveness during life cycle

The proposed approach consists of two main parts. The first, termed the War Room, is an elicitation exercise with the main stakeholders of the mission that serves to gather inputs regarding hypothetical consequences of cyber attacks that would be of most concern, as well as the context of mission goals, objectives, expectations, and procedures that would surround the identified consequences. By eliciting information from both the designers and users of the system, we attempt to minimize the disconnects between the design and actual implementation or use of the system. The second part consists of a modeling phase, which utilizes the information collected in the War Room. Specifically, the modeling phase utilizes systems and control theory to formalize the natural language outputs of the War Room and connect the abstract, mission-level information to concrete, hardware or software elements of the system. The resulting model can then be used for qualitatively identifying vulnerabilities and acting on them, or as a building block for more quantitative vulnerability and threat analysis.

The contributions of this section are:

- a new model-based analysis methodology that incorporates stakeholder perspectives to give a mission-centric viewpoint to enhancing the safety, security, and resiliency of a particular CPS;
- a modeling technique that captures the behavior of the CPS within its mission; and
- a concrete application to a real-world CPS and its corresponding mission.

OVERVIEW OF MISSION-CENTRIC CYBERSECURITY AND MODELING

The information carried out via the War Room analysis not only assists us in facilitating systematic requirements and model development but, also, allow us to secure system's more effectively by being *aware* of their mission-level requirements. For one, by going through the methodology above we are securing subsystems in a manner that is directly tied to important mission goals. Then, we can erect barriers in those subsystems that can assure, within tighter error bounds, the success of military missions, because we have addressed the possible insecure controls that can

lead to unsafe behavior. This benefit becomes more apparent when we deal with multiple complex system of systems (SoS) that coordinate with each other to achieve mission success.

Traditionally, there is no ``science'' to applying security as a structured assistant to mission success. Indeed, it is often true that the procedure of securing mission-critical systems is based upon an unstructured and ultimately random security assessment that might or might not lead to mission degradation (see policy lists). This is problematic because security should not be exercised for the sake of security but, in general, should be used as a tool to avoid possible transitioning to states that violate the system's expected service. Avoiding this transitioning to hazardous states is the *raison d'être* of security. Following that definition, any security measure that goes beyond providing assurance of safe behavior or any measure that doesn't adequately assure the safe behavior of the system during a mission is a loss of resources and, hence, can inadvertently be a hindrance in the command and control of military systems.

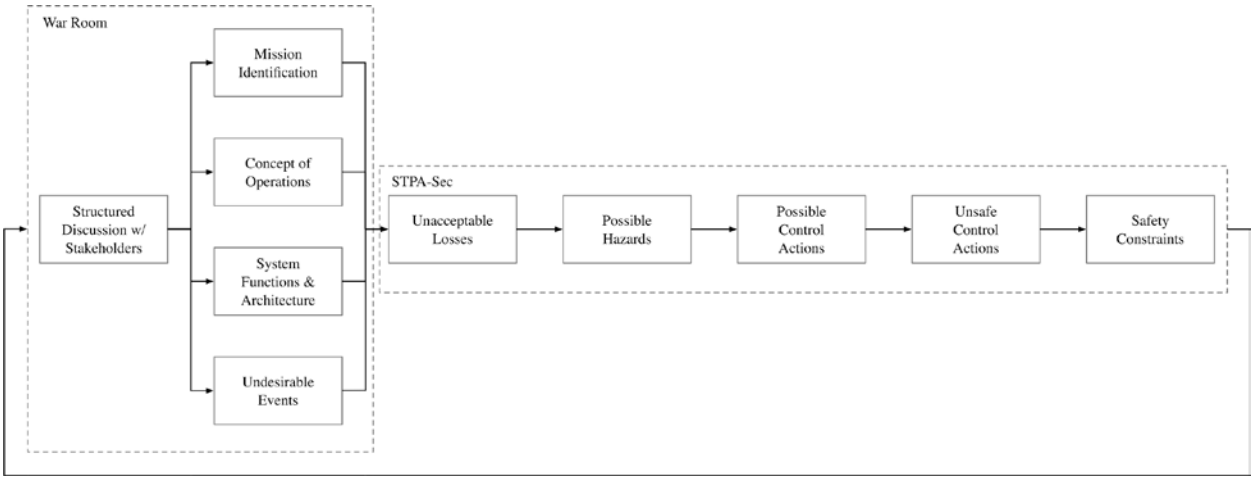


Figure 2 A concept view of how the War Room facilitates the STPA-Sec analysis.

THE WAR ROOM

Similar to the manner in which a group of military planners might organize an operation around vis a group meeting in a command post, our so-called War Room aims to gather as much information as possible about a mission and the use of a CPS within that mission. This includes the objectives, success criteria, time and location, human and material needs, and other similar information about a mission in general, as well as the particular roles that the component systems in a SoS configuration would play during that mission. This exercise is completed by an analyst team leading a structured discussion with a range of stakeholders relevant to individual CPS's and mission. The main products are outlined in Figure 3.2 and include a detailed, natural language description of the mission, a concept of operations (ConOps) specific to the CPS's use in that mission, a list of functions or components that are critical to mission-success, and insights about unacceptable, hazardous, or undesirable events or outcomes with respect to the mission. This information serves as the basis for the second step of our proposed methodology; however, it is extremely valuable on its own for guiding future cybersecurity solutions.

CONDUCTING THE WAR ROOM

A key tenet to our cybersecurity focused approach is bringing together the system's stakeholders that can better inform security solution designers regarding the use, operation, and requirements of the mission and, consequently, the desired behavior of the system itself. For a military CPS, these stakeholders include system designers, military commanders, military operators, maintenance technicians, and potentially other personnel that might hold pertinent information to the success of the mission.

By discussing with the stakeholders of the mission, cybersecurity solution providers are gaining an important understanding about the expectations, requirements, world-views, and interactions that each stakeholder has with the system. Since each stakeholder has a different role and area of expertise, the differing views on how the system operates and performs give more well-rounded insights and context to how a system will be used as part of a larger mission.

The cybersecurity analysis team is responsible for leading the discussion between the stakeholders. It is their duty to guide topics and ask specific questions that provide the information needed to construct a full model of the mission and CPS. The analysts should ensure that they have a clear understanding of the basic architecture, function, and purpose of the mission and CPS before moving on to the next phase of the discussion process.

The analysts should obtain a list of the mission goals, sub-goals, success criteria, and reasons for mission failure. Additionally, the analysts should consult with the mission planners on what may constitute an unacceptable outcome of the mission. For example, an unacceptable outcome could include collateral damage in an air strike mission or a failure to gather information in a surveillance mission. By listing out the unacceptable outcomes of the mission, we begin to develop a sense of mission critical functions, objectives, and actions.

After the analyst team gathers the more general information described above, the next step is to challenge the stakeholders' routines, expectations, and experiences with both the CPS and the mission. The analysts should ask questions, based on the previously gathered information, about what the stakeholder may do if a particular situation arises during the mission. For example, an analyst may ask the CPS operator what he or she might do if the CPS lost functionality during the mission. The purpose of asking these questions is two-fold: to get the stakeholders to think about how they may or may not be able to adapt to losses in functionality due to cyber events or other causes, and to further develop an understanding of the critical aspects of the mission and CPS. The answers to these questions can highlight potential oversights in the mission or the CPS, as well as further inform the construction of the model in the next steps of the methodology.

After the War Room is completed, the logs of the discussions contain vast amounts of information that is difficult to use on its own. Consequently, it is necessary to organize that information into

a more formal form, which can allow for direct analysis of the mission and CPS, in addition to providing the structure for later models used for automated vulnerability analysis.

STAMP AND STPA-SEC

To increase the interpretability of the information collected in the War Room, we propose using a modified version of STPA-Sec [Young, W., & Leveson, N. (2013, December)], which is itself derived from STAMP [Leveson, N. (2011)].

System-Theoretic Accident Model and Processes (STAMP) is an accident causality model that captures accident causal factors including organizational structures, human error, design and requirements flaws, and hazardous interactions among non-failed components [Leveson, N. (2011)]. In STAMP, system safety is reformulated as a system control problem rather than a component reliability problem—-accidents occur when component failures, external disturbances, and/or potentially unsafe interactions among system components are not handled adequately or controlled. In STAMP, the safety controls in a system are embodied in the hierarchical safety control structure, whereby commands or control actions are issued from higher levels to lower levels and feedback is provided from lower levels to higher levels. STPA-Sec is an analysis methodology based on the STAMP causality model, which is used to identify cyber vulnerabilities [Young, W., & Leveson, N. (2013, December)].

By using this framework, we are able to capture the relevant information from the War Room in a systems-theoretic model of the mission and the CPS. This model systematically encodes the unacceptable outcomes of the mission, the hazardous states that can lead to those outcomes, and the control actions and circumstances under which those actions can create hazardous states. This information can be modeled from the mission-level all the way down to the hardware and component level, which allows for full top-to-bottom and bottom-up traceability. This traceability allows us to evaluate the cascading effects of specific changes to hardware, software, the order of operations, or other similar events on the potential outcome of a mission. Consequently, we can then use this information to identify and evaluate vulnerable areas in a system and take steps to mitigate or eliminate those vulnerabilities.

CONSTRUCTING THE MODEL

The STPA-Sec model identifies how security issues can lead to accidents or unacceptable losses. In particular, the model outlines the behavior of the CPS and other actors within the overall mission and how that behavior can become unsafe. The general process for creating the STPA-Sec model is outlined in Figure 2.

The first step of building the model is identifying the mission to be performed, which was explicitly defined in the War Room. This statement takes the form of, ``The mission is to perform a task, which contributes to higher-level missions or other purposes." The specific language used here serves to succinctly and precisely outline the general purpose and function of the mission.

After defining the mission, the next step is to define the unacceptable outcomes or losses associated with the mission. For example, an armed UAV conducting a strike mission may have an unacceptable loss defined as any friendly casualties occurring. These losses or outcomes were either explicitly or implicitly identified and prioritized by the War Room stakeholders. For example, the failure to destroy a target may be less important to mission commanders than inflicting friendly casualties.

After defining the unacceptable losses, we define a set of hazardous scenarios that could potentially result in an unacceptable outcome. Some of these scenarios may have been described in the War Room; however, it is likely that many will be defined by the analysts on their own. For example, in the UAV mission and unacceptable loss described above, a hazardous scenario might be that friendly forces are within the targeting area. This on its own does not necessarily lead to the unacceptable loss of friendly casualties, but such an outcome is certainly a possibility if the munition is in fact launched. The set of hazardous scenarios does not have to be an exhaustive list; however, the analysts should strive to define a set of hazards that have a reasonable chance of occurring during a mission and also might be provoked by cyber attacks that corrupt relevant system software.

After defining the set of hazards, the analysts shall outline a functional hierarchy and the control actions that can be taken at each level during the mission. For example, in a typical mission, there might be three functional levels or actors: the mission planner, the system operator, and the physical system. Obviously, the defined functional levels depend on how the analysts define them and can vary depending on the system in question; yet this step is necessary as it allows us to scope the model to a reasonable degree of granularity. Next, the analysts define the control actions that can be taken at each level. A generic control loop is presented in Figure 3. In general, the control action at one functional level enacts a change onto a controlled process at a lower level via an actuator and then the controller receives feedback from the controlled process via a sensor. For example, a control action in the mission planning functional level could be defining a flight plan for an unmanned reconnaissance mission, and a control action at the operator level in the same mission might be commanding the vehicle to make a 30 degree turn to the north. The control actions and functional levels should be represented in a flow diagram that represents the planned order with respect to the mission. This will help analysts establish the ``baseline'' order of operations and procedures during the mission, which can be used later to analyze deviations from standard operating procedure.

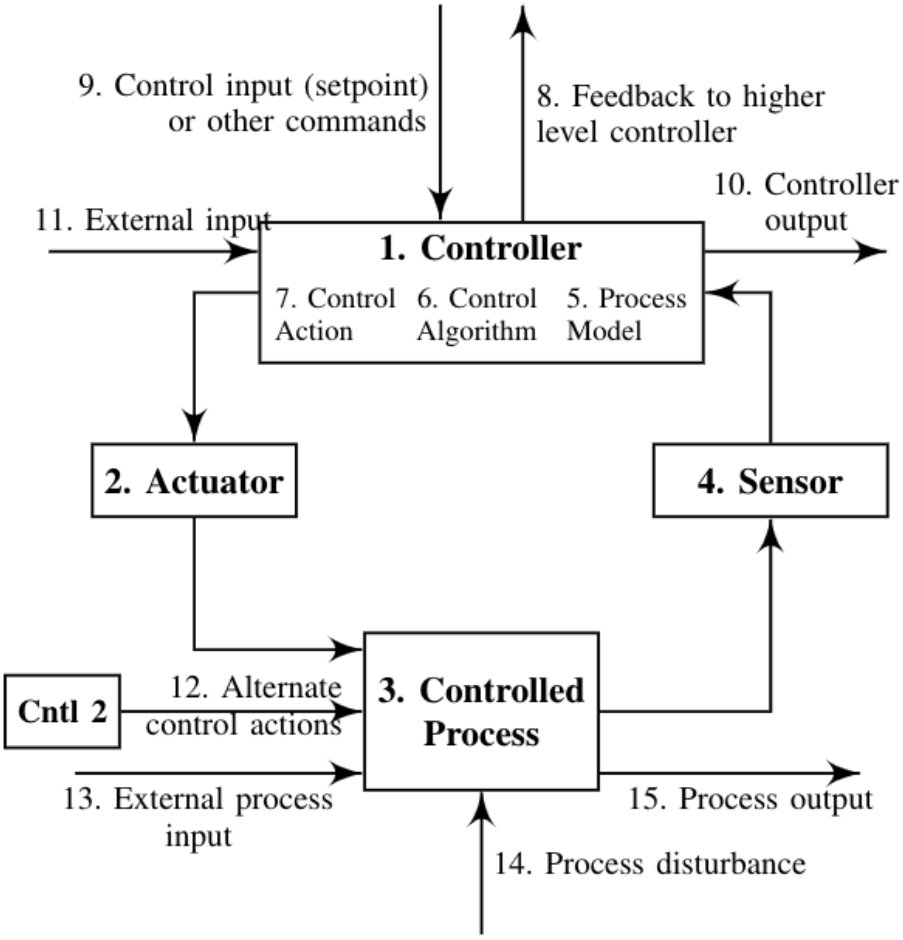


Figure 3 A control loop with generic entities

After defining the control actions within a mission, the next step is to define the circumstances in which a particular control action could be unsafe. These circumstances can generally be defined as being a part of one of the four following categories:

- not providing a control action causes a hazard;
- providing the control action causes a hazard;
- the control action is performed at the incorrect time or out of order;
- the control action is applied for too long or stopped too soon.

For example, in the reconnaissance mission described above, the turning control action could be unsafe when it is applied for too long and causes the aircraft to stall out. For each control action, the analysts should identify a circumstance for each of the four categories mentioned above in which the control action would be unsafe. These unsafe control actions are placed into a table, which then allows us to easily access information when needed during the next phase of analysis.

USING THE MODEL

At the beginning of this process, several unacceptable outcomes of the mission are identified and prioritized. The next step is to identify (a) how those losses may occur and then (b) how they can be avoided or mitigated. In the previous step, the circumstances under which control actions would become unsafe were identified. Now, those circumstances are used to derive security requirements and constraints on the behavior of the system. For example, a constraint in the UAV strike mission mentioned previously could be ``no fire munition command shall be issued when friendly forces are in the targeting area." These constraints are presumably already reflected in the operational procedures of the system.

Finally, we can identify causal scenarios using all of the previously defined information to determine how an unacceptable loss may occur. Using the UAV strike mission as an example, an unacceptable loss can occur when the operator issues a fire munition command when there are friendly forces within the target area because his or her sensors indicated otherwise. Such a scenario could feasibly be the result of a precedented Denial-of-Service Attack on the operator's sensor. By creating these causal scenarios, we seek to determine the most likely or most damaging pathways for potential security breaches. Furthermore, creating the STPA-Sec model helps identify the most critical components, features, or functionality in a system with respect to mission success. This information can then be used to guide which cyber-security or cyber-resiliency measures are implemented in the future.

APPLICATION OF APPROACH

This methodology is demonstrated by analyzing an intelligence-gathering mission using a small reconnaissance UAV. The results and outputs of the analysis are included in this section.

WAR ROOM FOR THE UAV MISSION

For the UAV mission, the War Room activity included two commanders who are responsible for planning and organizing the mission, two system designers with technical expertise for the UAV, and an analyst leading the War Room exercise.

At the beginning of the exercise, all stakeholders agreed on a general mission and system description before entering discussions. The description of the mission is as follows: ``The tactical reconnaissance mission is to obtain visual information on the activities or resources, or lack thereof, of an adversary within a particular geographical area so as to support other on-going or planned operations.'' The subsequent system description is defined as ``The tactical reconnaissance UAV carries an imaging payload to observe an area of interest and transmit the video feed to the appropriate decision-makers in support of other on-going operations in the region.'' The language chosen here is important for clearly articulating the purpose of conducting a mission with a particular CPS. The mission definition should highlight a need for a particular output and the system description should highlight how it can provide that output to the mission.

In this case, the reconnaissance mission requires information about enemy activities in a particular area, and the UAV can provide that information through visual images by loitering over the area.

Following the gathering of this general information, the commanders detailed the criteria for mission success and failure. Mission success would be completed via the collection and transmission of a clear, continuous video feed of the area of interest, regardless of what that video feed displayed. Mission failure would result from the failure to detect an actual threat in the area of interest. Furthermore, the military stakeholders also identified a small set of unacceptable outcomes. First and foremost, any loss of resources stemming from the lack of or inaccuracy of information collected during the reconnaissance mission is not usable. Additionally, loss of classified information or systems as a result of the loss of the UAV or any collateral damage that occurs as a result of the loss of the UAV, i.e. a crash, are unacceptable outcomes to the mission.

Finally, the analyst queried the stakeholders about how some abnormal scenarios might be handled. These questions generally take the form of ``what if this happens?'' Clearly, this is not an exhaustive approach to identifying possible courses of action in abnormal scenarios, but the purpose here is to get a general idea of how the stakeholders may react to losses of functionality in the CPS. For example, the military commanders indicated that the reconnaissance mission would fail if no visual information could be collected by the UAV and that information would need to be gathered by other methods. These other methods could involve sending in a team of reconnaissance troops; so obviously it would be preferable to avoid putting human lives at risk. The system designers indicated that if the UAV lost GPS service, then the integrity of the mission would be compromised, but not necessarily result in mission failure as the inertial navigation system would take over. In general, for this particular mission, the loss of video functionality directly results in mission failure; meanwhile, loss of other functionality, such as GPS navigation, can result in mission failure, but is not necessarily an immediate result. This information helps us prioritize unacceptable losses and hazards when we build the STPA-Sec model.

STPA-SEC MODEL FOR A UAV RECONNAISSANCE MISSION

The first step in building the STPA-Sec model is to define the mission and the CPS in the context of its role in the mission. This system and mission was defined as follows: ``A reconnaissance UAV is a system to gather and disseminate information and/or data by means of imaging (or other signal detection) and loitering over an area of interest to contribute to accurate, relevant, and assured intelligence that supports a commander’s activities within and around an area or interest.'' This statement is effectively a combination of the War Room definitions of both the UAV and the mission it performs. The next step is to identify the unacceptable losses that could occur during the mission. In this case, this information comes directly from the War Room. The unacceptable losses are defined in order of priority in Table 3.1. Given the tactical nature of this mission and the small size of the UAV, it is less vital to be concerned with the loss of the vehicle

itself, but rather the loss of potentially key intelligence from the inability to survey the area of interest.

Table 1 Unacceptable Losses for a UAV reconnaissance mission

Unacceptable Loss	Description
L1	Loss of resources (human, materiel, etc.) due to inaccurate, wrong, or absent information
L2	Loss of classified or otherwise sensitive technology, knowledge, or system/s
L3	Loss of strategically valuable materiel or personnel/civilians due to loss of control of system

Table 2 Hazards for a UAV reconnaissance mission

Hazard	Worst Case Environment	Associated Losses
H1- Absence of information	Imminent threat goes undetected	Manpower, materiel, territory, etc. **L1**
H2- Wrong or inaccurate information	Threat is incorrectly identified/characterized	Same as above **L1**
H3- Loss of control in unacceptable area	UAV is lost in enemy territory and suffers minimal damage in crash/landing	Compromise of critical systems, intelligence, and/or other potentially classified information or technology **L2, L3**

Table 3 Hazardous Control Actions

Control Action	Not Providing causes hazard	Providing Causes hazard	Incorrect Timing or Order	Stopped too soon or applied too long
CA 1.1- Designate area of interest	No information collected H1	Area is wrong or will not provide needed information H1, H2	Area designated no longer of use H1, H2	Area would be useful at another time H1, H2
CA 1.2- Specify Surveillance Target	Surveillance is not focused and provides too little or too much information H1, H2	Target is wrong or does not provide needed information H1, H2	Target no longer of interest or does not provide needed information H1, H2	Needed information occurs before or after surveillance H1
CA 1.3- Indicate type of intelligence needed	Gather too much or too little data to be useful H1, H2	Intel type is not appropriate for what is needed H1, H2	Type of intel collected at wrong time (i.e. SIGINT during time with no signals) H1, H2	Miss desired type of intel H1
CA 1.4- create rules of flight or engagement	UAV strays into inappropriate area H3	UAV cannot collect needed information H1, H2	Needed information not collected H1, H2	Needed information not collected H1, H2
CA 2.1- Designate surveillance strategy	Surveillance is ineffective, does not provide needed information H1, H2	Surveillance strategy is inappropriate, does not provide needed information H1, H2	Similar to left H1, H2	Similar to left H1, H2
CA 2.2 - Set flight parameters	UAV strays into inappropriate area H3	UAV has inappropriate field of view H1, H2	Similar to left H1, H2	Similar to left H1, H2
CA 2.3- Start Process (begin surveillance)	Information not collected H1	Inappropriate information collected H1, H2	Needed information not collected H1, H2	Same as left H1, H2

CA 2.4- Make maneuver command	UAV strays into inappropriate area, field of view not adjusted appropriately H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3
CA 2.5- End process	?	Needed information not collected H1, H2	Same as left H1, H2	Same as left H1, H2
CA 2.6- Make data collection command	Needed information not collected H1, H2	Same as left H1, H2	Same as left H1, H2	Same as left H1, H2
CA 3.1- Compute, translate, or interpret command	Stable flight not achievable, field of view not appropriately adjusted H1, H2, H3	Stray into inappropriate area, field of view not appropriate H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3
CA 3.2- Send Signal	Same as above H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3
CA 3.3- Interpret Feedback	Same as above H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3
CA 3.4- Determine orientation and location	Same as above H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3
CA 3.5- Report Information	Operator does not get information, same as above H1, H2, H3	?	Cannot fly properly H3	Same as left H3
CA 4.1- Move control surface	UAV does not avoid inappropriate area, field of view not adjusted properly H1, H2, H3	UAV enters inappropriate area H1, H2, H3	UAV fails to avoid inappropriate area H1, H2, H3	UAV temporarily enters inappropriate area H1, H2, H3

CA 4.2- Take picture or collect data	Needed information not collected H1, H2	Wrong information collected H1, H2	Needed information not collected H1, H2	Needed information not collected H1, H2
CA 4.3- Send data/feedback	Information not supplied to controller H1, H2	Wrong information sent to controller H2, H3	Information not sent to controller at correct time H1, H2	Inadequate information sent to controller H1, H2
CA 4.4- Send feedback	Same as 4.1 & 4.2 H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3	Same as left H1, H2, H3

Table 4 Safety Constraints for a subset of the control actions

Control Action	Safety Constraint
CA 1.1- Designate area of interest	The mission planner shall always clearly define the area of interest to align with any future mission that the for which the reconnaissance is needed
CA 1.2- Specify Surveillance Target	The mission planner shall indicate as specific a target as possible for the reconnaissance
CA 1.3- Indicate type of intelligence needed	The mission planner shall designate a specific type of intelligence that the mission is going to collect
CA 1.4- Create rules of flight or engagement	The mission planner shall indicate a specific set of rules of engagement to prevent confusion
CA 4.1- Move Control Surface	Control surfaces shall only move upon receiving authentic commands from the flight control system
CA 4.2- Take Picture or Collect Data	Data collection shall only occur upon authentic command from the operator

CA 4.3- Send data/feedback	The component shall relay collected data or send feedback to the appropriate monitors at regular intervals

Next, following the work flow in Figure 2, we identify a set of hazards, the worst-case environment for that hazard to occur in, and the unacceptable loss that could result from that hazard. These hazards are defined in Table 2. The three hazards listed were determined to create the greatest threat of resulting in an unacceptable loss. The War Room indicated that the information collected during this mission is critical for mission success; therefore, the top hazards relate to the absence or unreliability of information.

The next step is to identify the generic control actions that can be taken at different functional levels in the system in order to provide causal paths to a particular hazard. For this mission, there were five functional levels defined: Mission-level requirements or plans, the human operator or pilot, the UAV autopilot system, the control servos and imaging payload, and the physical environment in which the UAV operates. At each level, there are a set of control actions that can be taken to influence the behavior of the surrounding levels, apart from the physical environment, which only provides disturbances to the control structure. This generic structure is represented in Figure 4. For example, at the mission-level a commander will designate the area of interest for the reconnaissance mission, which feeds into the actions that the pilot takes to satisfy that requirement, and so on. For each control action, there is a scenario in which one of the four types of unsafe control actions creates at least one of the hazards identified in Table 2. A subset of these control actions and circumstances are outlined in Table 3.

Now that we have identified the control actions available in the system and the conditions under which they create hazardous scenarios, we can identify a set of constraints that can be applied to the behavior of the system to limit the possibility of a hazardous scenario leading to an unacceptable loss. The constraints defined for the control actions outlined in Table 3 is presented in Table\vref 3.4.

In addition to the constraints that should be applied on the system, analysis of the STPA-Sec model identifies areas that should receive the most attention in order to increase security and resiliency against cyber attacks that can produce unacceptable mission outcomes. For the UAV reconnaissance mission identified in this example, the most pressing unacceptable outcome relates to military commanders not receiving vital information about potential enemy activity within an area of interest. In this case, the integrity of the video feed coming from the UAV should receive top priority. Developing and evaluating measures for ensuring integrity of the video feed (or assuring that the system can identify when integrity has been lost) is outside of the scope of the current project.

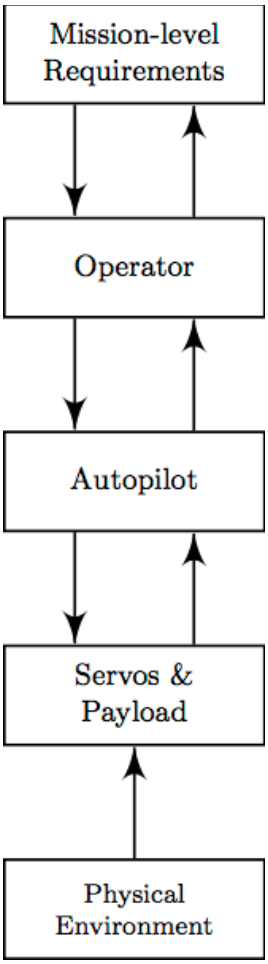


Figure 4 A hierarchical controls model that defines the expected service of a UAV. Each level is defined by a generic control structure. Inadequate control in each level can cause an adversarial action to degrade the expected service and produce a hazardous state.

RELATED WORKS

Cybersecurity generally follows a software-oriented perspective. Mead and Woody present state-of-the-art methods for software assurance in Mead, N. R., & Woody, C. (2016), and focus on integrating security earlier in the acquisition and development cycle of software. Furthermore, Mead, Morales, and Alice describe an approach that uses existing malware to inform the development of security requirements in the early stages of the software lifecycle [Mead, N. R., Morales, J. A., & Alice, G. P. (2015)]. This approach seeks a similar product to the one presented in this paper; however, it follows the standard, bottom-up approach of identifying threats and generating solutions based on those threats. These techniques work well for IT software systems, yet are insufficient for cyber-physical systems. Hu asserts in Hu, F. (2013) that these cybersecurity approaches are not effective for CPS as an attack on a physical system is not necessarily detectable or counteracted by cyber systems. Burmester et al define a threat modeling framework specifically for CPS that takes into account the physical component of CPS

that many other methods do not [Burmester, M., Magkos, E., & Chrissikopoulos, V. (2012)], yet this approach still relies on historical threats to identify vulnerabilities.

STPA-Sec [Young, W., & Leveson, N. (2013, December)], however, aims to reverse the tactics-based bottom up approach of other cybersecurity methodologies. While we seek to address the same issue, our implementation of STPA-Sec differs in two key areas. First, the implementation presented by Young and Leveson is a methodology to be performed by analysts on their own, whereas our implementation is informed explicitly by mission and system stakeholders via the concept of the War Room. This aids the STPA-Sec analysis by minimizing the chance of outputs not matching the perspectives and experiences of the stakeholders. Second, the approach presented in this paper introduces a mission-aware viewpoint to the STPA-Sec analysis. That is, one could have the exact same CPS in a completely different mission context, and would potentially want to choose different security solutions. The incorporation of the mission into the analysis scopes the security problem above the cyber-physical system level, which both opens up possibilities for potential vulnerability solutions, and motivates the choice of security or resiliency-based solutions.

DISCUSSION & CONCLUSIONS

This paper presented a systems approach to augmenting security and resiliency in cyber-physical systems. This framework is based on a top-to-bottom identification of unacceptable losses or outcomes to a particular mission that the CPS performs and examines how the paths to those outcomes can be avoided. We have shown an application of this approach to a hypothetical tactical reconnaissance mission using a small UAV and generated a set of constraints that should be present in the behavior of this example system to avoid pathways to unacceptable outcomes. In addition, this approach identifies the areas most critical to mission success as starting points for future implementations of security or resiliency solutions.

A future direction based on the findings of this work includes implementing the identified system constraints on model and formally checking that they can avoid unacceptable losses to the mission. Additionally, this work could be extended by closing the loop and testing security or resiliency solutions' effects on the behavior of the system in its mission. This would allow security and resiliency solutions to be evaluated based on their cost, complexity of implementation, and effectiveness at preventing unacceptable mission outcomes.

Through this work, we have identified an approach to reversing the traditional bottom-up nature of other security methodologies based on a mission-aware viewpoint. This approach recognizes that security is a hard and complex problem, but seeks to manage the costs and complexity of increasing security and resiliency by focusing on avoiding unacceptable losses rather than reacting to threats as they appear.

FORMAL MODEL DEVELOPMENT

A major challenge in Cyber-Physical Systems (CPS) [National Research Council. (2015)] is the assessment of the system's security posture at the early stages of its life cycle. In the defence community, it has been estimated that 70-80\% of the decisions affecting safety and security are made in the early concept development stages of a project [Corbett, J., & Crookall, J. R. (1986), Frola, F. R., & Miller, C. O. (1984), Kutz, M. (Ed.). (2015), Saravi, M., Newnes, L., Mileham, A. R., & Goh, Y. M. (2008)]. Therefore, it is advantageous for this assessment to take place before lines of code are written and designs are finalized. To allow for security analysis at the design phase, a system model has to be constructed, and that model must reasonably characterize a system as well as be sufficiently detailed to enable us to match attack vectors mined from databases. Matching possible attack vectors to the system model facilitates detection of possible security vulnerabilities in timely fashion. One can then design systems that are secure by design instead of potentially having to add *bolt-on* security features later in the process, an approach that can be prohibitively expensive and limited in its mitigation options. Consequently, employing a model reduces costs and highlights the importance of security as part of the design process of CPS. We propose a model encoded in the Systems Modeling Language (SysML), a graphical object oriented modeling language [Hause, M. (2006, September)].

Our justification for using a solution based on SysML is twofold: it facilitates the implementation of any changes to the design of CPS and is a tool familiar to and often used by systems engineers. However, as we do not want to limit the principle ideas of the model to SysML we show that the model is transformable to a graph structure, thus demonstrating that it is agnostic to the particular modeling language or tool used.

In order to accurately characterize the system with respect to its associated security audit, the model requires a taxonomic scheme consisting of predefined categories. To develop this schema, we investigate attack vector databases and examine their entries and their intra and inter connections. To represent the reality of the system, the chosen categories and their corresponding attributes need to correctly capture the cyber form as well as the interactions in the system model in such a way that the model is sufficient and can be used to find attack vectors from open databases.

Since the model drives the cyber-vulnerability analysis, it needs to reflect relevant cyber-oriented information required to match possible attack vectors within the aforementioned categories. This additional information is encoded in the attributes of a subsystem and normally comes from preexisting design documentation, subject matter expertise, and requirements documentation. The amount and level of specificity that needs to be embodied in the attributes of each component depends on whether it matches the natural language that describes the attack vectors within the vulnerability databases. Therefore, the attributes need to represent the system's hardware and software composition so that they match possible entries in the databases. Only then can a system model become ``cyber aware,'' enabling us to infer possible vulnerabilities in the system's architecture and propose preemption and mitigation strategies.

Construction of a model with the characteristics described above gives rise to two main challenges. The first is the complexity in creating a complete schema that is be able to capture---through the attributes---the necessary detail to represent the cyber aspects of an actual system. This first challenge also needs to take into account the importance of reasonably allowing the maintenance of any given CPS model that contains both cyber attributes and non-cyber attributes. The second is the difficulty in coordinating the available vulnerability data, the hierarchy within the databases that provide that data, and finally, the way in which the information is captured in their entries.

THE CHALLENGE OF CAPTURING A SUFFICIENT SET OF ATTRIBUTES

Indeed, there are tradeoffs between specificity and the number of attributes we can impose on a modeling methodology before it becomes impractical. This tradeoff can be illustrated by considering the extremities of model fidelity in Figure 5. On the one side of the spectrum (Low-Low) the model does not correctly characterize an actual system and its potential vulnerability to existing threats, whilst on the other side of the spectrum (High-High) the model requires a prohibitive amount of modeling effort.

Being model-driven in the context of cybersecurity has an additional requirement beyond ensuring fidelity to the real system. It also requires integrating the attributes taken from a design specification documentation, such that an analyst can match possible attack patterns to the model. This challenge is partially solved by understanding and using several repositories of attacks and vulnerabilities, which is described further in a later section. However, it still does not guarantee that possible query words---word lists that are used to associate potential vulnerabilities with a given set of attributes---will produce attack vectors applicable to that subsystem. Therefore, we need to capture the design documentation only to a point where there is agreement between two perspectives; (i) fidelity to the system's behavior and structure and (ii) the system's corresponding attack vectors.

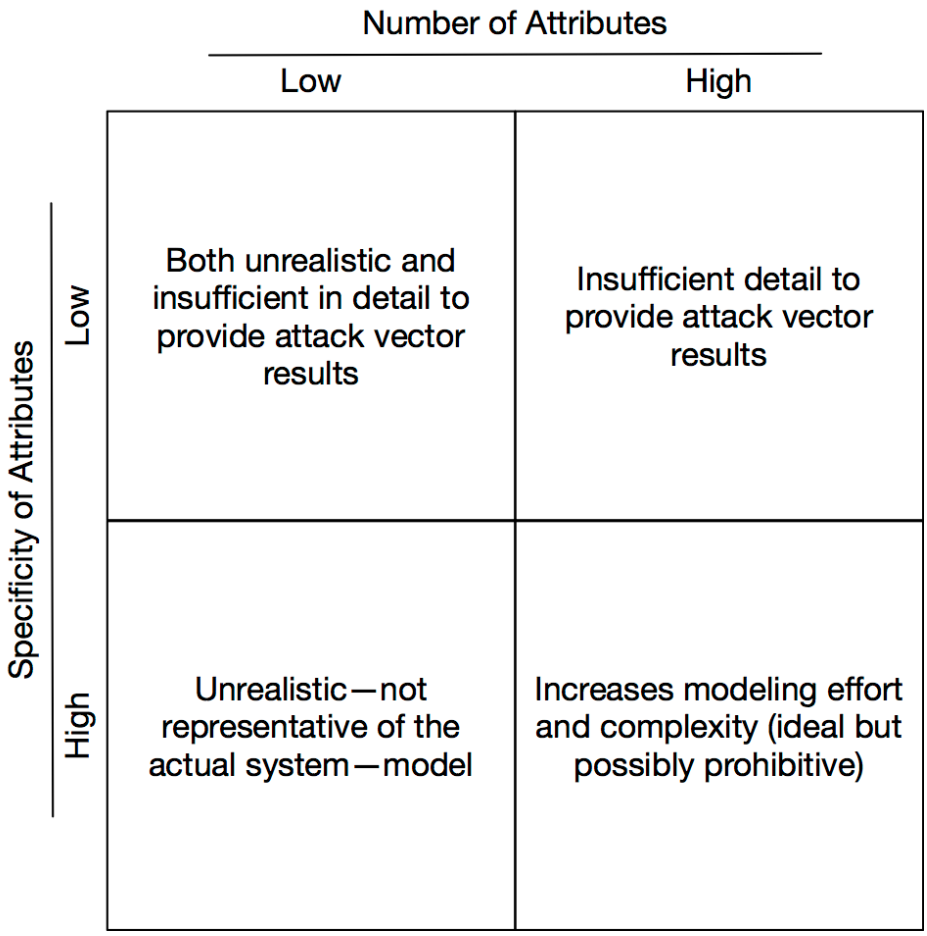


Figure 5 The fidelity of the attributes describing a CPS has to achieve a balance between the amount of information that is captured in the model and the difficulty of producing it. At the extremes, the attributes can either be uninformative and incomplete or too detailed, requiring a prohibitive amount of modeling effort. Part of the challenge to being model sufficient is producing a sound, well-formed model for cyber assessment that can be practically utilized by systems engineers.

CHALLENGE OF UNDERSTANDING DIVERSE VULNERABILITY DATA

Unfortunately, there is no single repository that contains all possible exploitation techniques or vulnerabilities that can apply to complex systems like CPS. If there was one, it would have to span across several domains---e.g., embedded devices, networks, humans in the loop---and have multiple levels of specificity for each entry in order to match every element in the system model. We leverage several databases with the goal of addressing all domains and at different levels of specificity, thus leading to new insights about the system's security posture. We then need to understand the different possible repositories, what information they contain, how they capture that information in natural language and what the targeted scope of each database is. By gaining that understanding, we extend the schema to incorporate data from diverse resources and ensure a thorough and rigorous security assessment.

Finally, the level of abstraction of each repository has to match the attributes in the model. This requires in the least a database that matches the attributes in the model. Ideally, however, there would be several databases that provide layers of abstraction. This is the case with a set of databases that are hierarchical and interconnected, e.g., CAPEC[4], CWE[5], CVE[6].

For this set of databases, CVE can match to the model attributes (since its vulnerability entries resides at a lower level of abstraction than the other two) and then be used to infer possible classes of attacks by mapping CVE entries to their corresponding, higher-level, CWE and/or CAPEC entries. The utility of this hierarchy is also important when reasoning about the threat space associated with the system. By abstracting the individual exploited vulnerabilities to more general weaknesses and/or patterns, this approach reduces the amount of information a cyber analyst has to parse and reason with when they inspect a complex system model for applicable attacks.

STATE OF THE ART

Model-based techniques for assessing cybersecurity have been at the forefront of academic research in CPS. These traditionally stem from dependability and safety analysis. Nicol, D. M., Sanders, W. H., & Trivedi, K. S. (2004) have stated the need for model-based methods for assessing security that come from the general area of dependability. Further, Chen et al. (2013, December). have proposed a model-based graph oriented analysis technique for assessing a system for acceptable safety based on a workflow. Kopetz, H. (2011) presents the notion of categories of interfaces to model real-time systems. Davis et al. (2015) present a framework that extends the notion of dependability to include possible security violation for the power grid that utilizes state estimation and is evaluated in a simulated model of the power grid. More recently, Brunner, M., Huber, M., Sauerwein, C., & Breu, R. (2017, July) proposed a combined model for safety and security that is based on Unified Modeling Language (UML) diagrams. These models reside in a higher level of abstraction than our proposed model; they do not contain the structure of the system, do not consider evidence-based security assessment, assessment based on previous reported vulnerability data, and the evaluation targets certification of policy standards.

In general, little work has been done to determine whether a model contains the necessary attributes stemming from the design documents and encoded through a structural model to be used in evidence-based cyber-vulnerability assessment. Even less work has been done in targeting the model sufficiency of CPS and how that is used at the early stages of the design process. For the cybersecurity assessment of CPS, no standard rule-of-thumb, or otherwise generally accepted procedure has been established.

[4] Common Attack Pattern Enumeration & Classification, capec.mitre.org
[5] Common Weakness Enumeration, cwe.mitre.org
[6] Common Vulnerabilities and Exposures, cve.mitre.org

CONTRIBUTIONS OF THIS SECTION

The central contribution of this paper is the characterization and definition of the problem of agreement between system model and historic attack vector databases and tackling the difficulty of matching the two when it comes to CPS. Therefore, this research presents solutions to the two main challenges outlined above. First, it shows how to capture the relevant information within those schema categories based on design documentation that preexists the model. Second, it demonstrates how to appropriately handle historic cyber-vulnerability data to identify possible vulnerabilities in the system's design. Both challenges can be solved by methodically constructing a model through a predefined taxonomic scheme. Toward those objectives, the model is built based on the following insights:

1. a necessary understanding of historic information is needed to match against a system description;
2. a system description needs to be evaluated as being realistic and it is characterized by attributes to an extent that can match possible attack vectors; and
3. agnosticism toward modeling language or tool.

This paper focuses on model sufficiency with respect to cyber vulnerabilities, with the explicit recognition that these vulnerabilities can give rise to unsafe or undesired behavior in the overall, coupled CPS. Moreover, this paper demonstrates how cyber vulnerabilities propagate to physical system behavior but it is outside the defined scope to analyze the physical behavior and/or determine whether it is (un)safe or (un)desirable.

To assess the sufficiency of the model we present a model of a generic Flight Control System (FCS) and assess the possible security violations of two system components using open vulnerability databases.

A TAXONOMIC SCHEME FOR CPS ATTRIBUTES

Our main objective is to construct a general purpose taxonomic scheme that can be used to characterize the cyber components of a CPS and their interactions, for the purpose of relating to attack vectors. To this end, we use preexisting design specification documentation to describe the attributes of the cyber components and encode this information in the model. To methodically achieve that we first present definitions for cyber component, cyber attribute, attack vector, evidence, and taxonomic scheme. These definitions provide common ground on the relatively generic term ``cyber'', which is used in several contexts and, therefore, can hold different meanings.

PRIMITIVES

Our model of CPS makes a distinction between cyber and physical components. Caution is necessary because the components of a CPS can reside in between the cyber and physical realms but their behavior and form can be described by either. This paper is intended only to identify the minimum set of attributes necessary to assess a CPS's cybersecurity posture.
% GB: Not sure this is necessary, the sentence above actually makes the point.
%Related work~\cite{if-we-want-to-cite-something} explores how cyber vulnerabilities can then lead to physical hazards.

> **Definition 1** A *cyber component* of a cyber-physical system is any device that is programmable and whose associated computation controls physical quantities, e.g., velocity, altitude, etc.
>
> **Definition 2** An *attack vector* is a specific description of an attack on a given subsystem. It presents the features of the exploited vulnerability, the privilege access level required for an attacker to perform the attack, and the steps to perform it.
>
> **Definition 3** A *cyber attribute* defining a subsystem of a cyber-physical system represents possible specification of behavior, form, or structure. Hence, the set of attributes produce a architectural definition of the subsystem and represents the part(s) of the model that maps to possible attack vectors.
>
> **Definition 4** *Evidence* is all instances of historic vulnerability data---attack vectors---that can be mapped to a cyber component through any cyber attribute or a combination of cyber attributes.
>
> **Definition 5** A *taxonomic scheme*, or schema, is a discrete set of categories that can capture the structure of any cyber component in a cyber-physical system to produce evidence.

Following the above definitions, the taxonomic scheme should inform the following questions:

1. What is the subsystem?
2. How is it implemented?
3. Who does it talks to?
4. Why is it there?

The first question informs the model about the identity of the subsystem. The second question provides the design details, used by the model to characterize a possible real subsystem. The third question identifies the required interactions between the subsystem, ensuring that the composition of the full system can provide its expected service---an important aspect of CPS as indicated throughout the literature [Sun, C., Ma, J., & Yao, Q. (2016).]. The fourth and last

question addresses the function of the subsystem and, therefore\ informs about its relative criticality to the overall behavior of CPS.

Answering the above questions sufficiently and constructing a taxonomic scheme directed by them allows us to base our analysis in methodical reasoning. Furthermore, by utilizing methodical reasoning we are able to view the threat spaces of a system holistically and take into account that a single segregated component acts differently than when it coordinates with other subsystems to compose a CPS. By using this approach, one can be better informed about the overall system security posture than by analyzing just the system's components individually.

REALIZATION OF THE TAXONOMIC SCHEME

Using the above questions as a guide and being cognizant of the intrinsic structure and specificity contained in open vulnerability databases, the following taxonomic scheme composes the structure of any CPS and can assist in producing evidence:

Operating System Since CPS is composed of hardware and software, it is important to be aware of the system software hosted, such as Real-Time Operating Systems (RTOS), executives, debuggers. In some cases, the embedded devices may be entirely hardware, a common term for which is ``programmed on bare metal.'' Knowing this information--that is, CPS running an embedded operating system--informs on possible vulnerabilities, e.g., bugs in the Linux kernel.

Device Name The specific naming of a subsystem can assist in finding device-specific vulnerabilities.

Hardware The decomposition of the specific device to its possible exploitable hardware elements, e.g., what chipset is on-board.

Firmware The possible firmware and corresponding drivers necessary to run the device.

Software In the event that the subsystem runs an RTOS it is of importance to know the possible software that is installed and can potentially introduce further vulnerabilities.

Communication Any cyber or physical interaction the CPS must implement in to provide its expected service.

Entry Points All possible accessible entry points to the system. This attribute allows us to filter components that are part of the attack surface.

ATTRIBUTES

In accordance with the taxonomic scheme above, a minimum set of attributes is presented in Table~\ref{tab:attributes}. The example comes from an NMEA GPS and its corresponding attributes stem from design documents and are refined using data sheets. Further refinement of those attributes is allowed. This dissemination is mostly based on the information provided in the design requirements documents but can also include information provided by subject matter experts.

> Insert table with: A GPS example of the minimum set of attributes necessary to create a sound, well-formed model of CPS. These attributes need to be used for any given subsystem that is pertinent to its expected service. The matching of attack vectors derives from the attributes specified in this table.

SYSML MODEL

Consider, for example, an Unmanned Aerial System (UAS) that is used to assist in search and rescue operations where minutes (or even seconds) count. In this domain, losing a UAS is certainly going to risk longer mission times and can potentially lead to an unsuccessful search and rescue operation. For these types of safety-critical missions, it is essential to assess the security posture of a given FCS design, so that a threat actor cannot interfere with safety-critical operations. For that reason we construct and evaluate an FCS model for possible security violation in an evidence-based fashion.

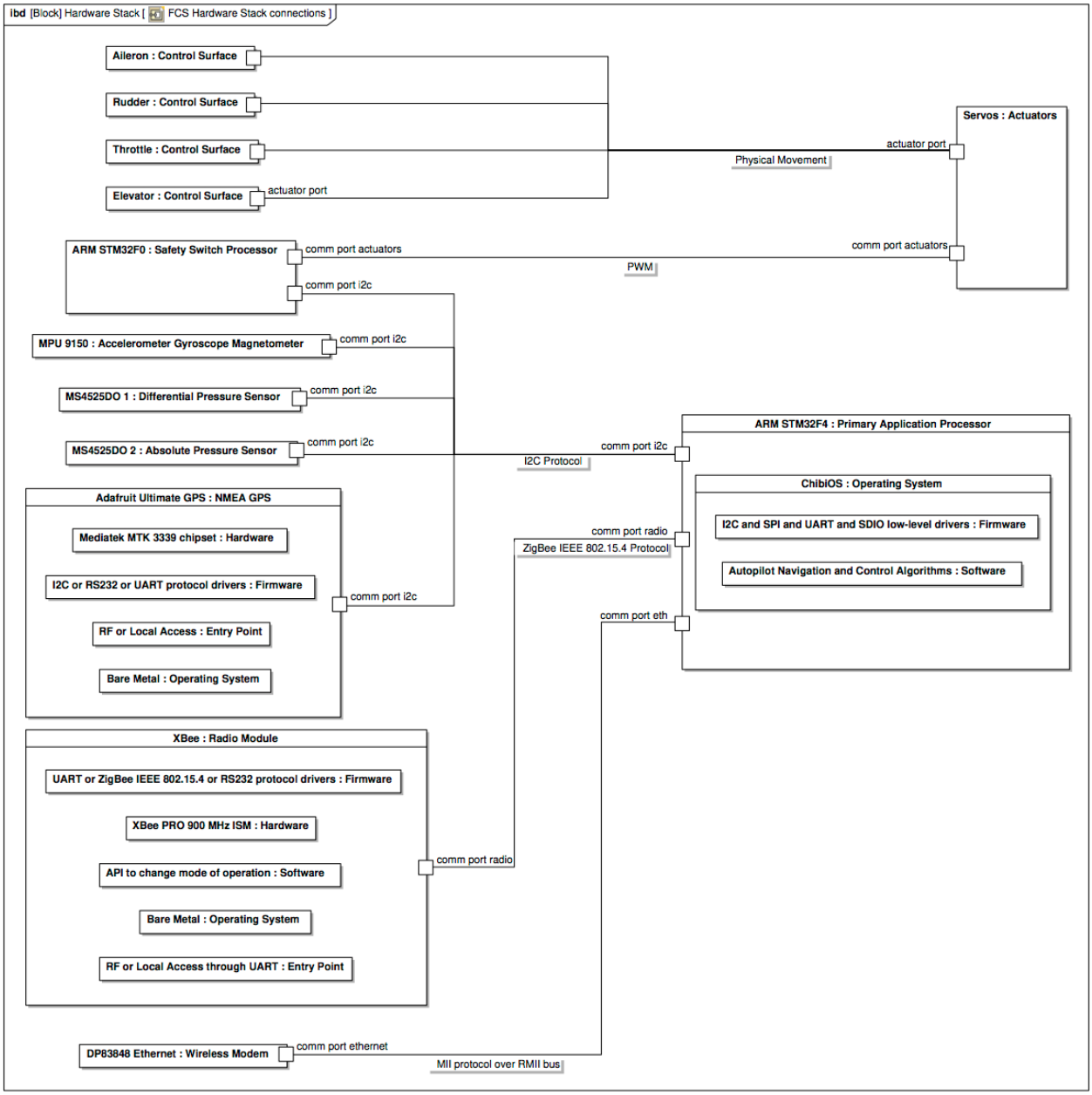


Figure 6 The Internal Block Diagram (IBD) SysML model of the Flight Control System (FCS) with all the necessary attributes to characterize the primary application processor, the NMEA GPS, and the radio module. This model can inform about possible architectural changes at the early stages of design to avoid using components that have reported vulnerabilities or clearly mitigate against these vulnerabilities through better informed requirements. As such, we are able to construct systems that are secure by design.

In the most general sense an FCS implements all the capabilities and control surfaces needed to operate an autopilot [Ward, G. L., Bakirtzis, G., & Klenke, R. H. (2014), Ward, G. L. (2014)]. This autopilot is used to fly UAS and is usually controlled through a ground control station. An FCS is composed of several compromisable cyber subsystems whose intended function is to modify and control physical parameters, e.g., control of engine throttle for speed, changing the direction

based on a navigational goal by controlling the aileron, etc. By this definition, an FCS is a CPS and prime example of analysis given its ubiquity and utility in many domain areas. An FCS is safety-critical system, in the sense that if there is a hazard, either artificial through a cyber attack or by natural faults, it can cause severe consequences.

The model of the FCS is encoded in an Internal Block Diagram (IBD) (Figure 4.2). An IBD representation is used to define a system's structure. Traditionally, this model contains only generic representations of subsystems, e.g., power system, engine sensor, magnetometer and general information about flows of information, e.g., energy, command, sensor measurement. The taxonomic scheme described in the previous section adds specific implementation information that assists cyber analysts in finding possible vulnerabilities and, consequently, associated attack vectors. This specific implementation information is encoded using part properties for component attributes, e.g., what type of GPS, and connectors for interaction attributes, e.g., using the I2C protocol.

Part properties encode further attribute information in a manner that is easy to parse by cyber analysts. Part properties, as the name implies, are attributes that can further characterize an IBD block and take the form of <part name> : <type>. Additionally, they can be decomposed to further part properties to make general categories of attributes---this can be seen by the decomposition of the *Operating System* type in the *Primary Application Processor* to further part properties (Figure 4.2). Moreover, part properties can construct a new IBD by themselves to define the connectivity between components in a collection of part properties. This is useful in more complex systems where the connections might have different levels of abstraction. Part properties define the structure and composition of the system.

Finally, connector types define source and target relationships as well as the type of interaction, digital protocols, analog inputs, or possibly physical actions. For example, in Figure 4.2 the connection between the *Safety Switch Processor* is digital and uses Pulse Width Modulation (PWM) commands to move the servos. The servos then provide the physical pull to either open the throttle further---to gain velocity---or change the direction of movement of the aircraft by controlling the aileron.

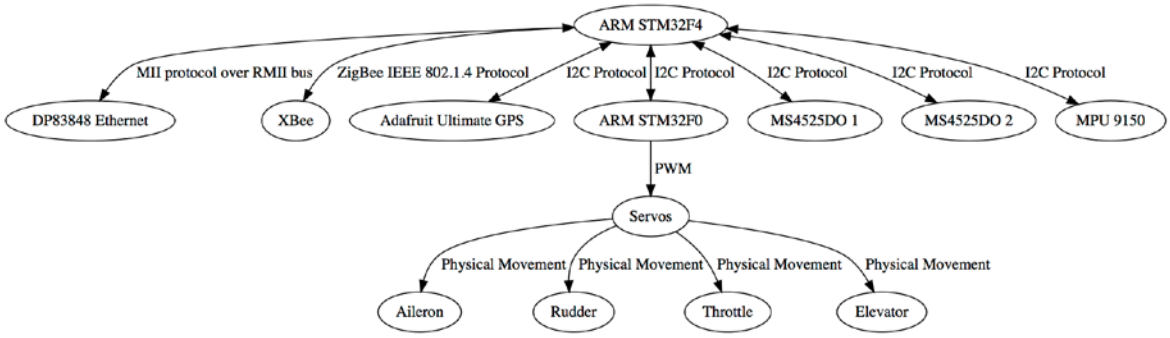


Figure 7 The Internal Block Diagram (IBD) (Fig.~\ref{fig:ibd}) of the Flight Control System (FCS) maps directly to a graph structure (the part property type is omitted for visualization purposes). This allows us to extract the elements in an internal block diagram without loss of any information vital for cybersecurity assessment. The graph structure can be further input to other analysis techniques and provide a schema for the topological definition of a Cyber-Physical System (CPS). In this instance the vertex attributes can be accessed through the GraphML specification even though they are not visualized here.

MODEL TRANSFORMATION

A model transformation is used so that the modeling methodology presented in this paper is agnostic to specific modeling tools and languages. To achieve this transformation, we construct a generic formalism for SysML IBD based on graph structures. The formalism is used as a basis for a tool which extracts the information from the IBD and encodes it in GraphML. GraphML is based on XML and consists of the de facto schema for sharing graphs [Brandes, U., Eiglsperger, M., Lerner, J., & Pich, C. (2013)]. Transforming SysML models to a graph should allow, in the future, the model to be analyzed with a variety techniques. This is potentially beneficial because SysML has limited verification capabilities and requires a specific modeling methodology to produce validation results. The following formalism assures that all SysML information and their corresponding properties are not lost in the transformation to GraphML.

FORMAL SEMANTICS OF INTERNAL BLOCK DIAGRAMS

Computing networks are typically reasoned about using graph structures, where the nodes represent assets and the edges represent their immediate connections. This is no different for CPS [Weaver et al. (2013, November)]. This paper extends the definition of assets to encompass every subsystem in a CPS, which comprises of more than the computing systems, including but not limited to imagery payload, actuators, sensors and their data links to the computing systems. We formalize these definitions, which are initially encoded in a SysML IBD, using standard graph notation below.

Definition 6 An *internal block diagram* is a graph *G := (V, P, src, tgt, A)*, where *V* is the set of vertices of *G*; *P* is the set of ports of *G*; *src*, *tgt*: $P \rightarrow V$ functions source and target for *G* respectively, and *A* is the set of attributes of *G*. *V* represents the components of a cyber-physical system, *P* the inputs and outputs corresponding to the components, *src*, *tgt* the directionality of the possible cyber or physical interactions between components, and *A* the associated descriptors for a given vertex or connection.

An example transformed system graph is depicted in Figure 4.3 where not all information is necessarily visualized but is encoded in the GraphML format and can be accessed programatically.

Definition 7 A *part property* in an internal block diagram is a function $attr_v: V \rightarrow A_v$, where *V* is the set of vertices of *G* and A_v the set of vertex attributes, such that $A_v \subseteq A$.

Consider a single mapping for the attributes of the NMEA GPS, (Table~\ref{tab:attributes} and Figure 4.2):

$attr_v$ (GPS) $\rightarrow$ {Bare Metal, Adafruit Ultimate GPS, Mediatek MTK 3339 chipset, ... , RF}

Definition 8 A *connector* in an internal block diagram is a function $attr_p : P \rightarrow A_p$, where *P* is the set of ports of *G* and A_p the set of port attributes, such that $A_p \subseteq A$.

For example, the tuple of vertices below passed into the port attribute function will provide the edge-specific attributes for that tuple: $attr_p$({GPS, STM32F4}) $\rightarrow$ {I2C Protocal}.

Given the above definitions, we transform the SysML model to the neutral GraphML format and can further use it as input to other analysis techniques, including finding attack vectors.

MATCHING POTENTIAL ATTACK VECORS

To evaluate the model we find and map applicable attack vectors for subsystems of the FCS model described above (Figure 6 and Figure 7). This analysis is based on analyzing open vulnerability databases to find possible attacks and provide possible design mitigation strategies. For example, given a component with an associated set of attack vectors, one can then assess the risk and potentially find another component that provides the same expected service without any reported vulnerabilities.

Toward the evaluation of the model's fidelity we find potential attack vectors for the subsystems of the FCS model using cve-search[7]. This online database not only provides possible CVE entries that are applicable based on the query strings produced by the model's attributes, but can also relate to higher levels of abstraction, e.g., CWE or CAPEC, to further understand the possible impact of a given attack.

We assume that a system is vulnerable if any single attack vector from the databases maps to any single attribute of the system model. For example, if an attack vector targets Operating System `A' with Driver `B', we consider it a vulnerability even if the model includes only `A' or `B'; it does not have to contain both, even if the attack pattern specifies them together. This assumption is reasonable because a large number of vulnerabilities that are reported have to do with systems that are popular and widely used, e.g., the Android operating system and corresponding drivers. This assumption allows for extrapolation from such (specific) reports to better understand the security posture of the model where, at the moment, no embedded system vulnerability database exists.

Furthermore, it is uncommon for users to update their embedded devices, and this assumption still allows the analyst to take into account vulnerabilities that may have been fixed in newer versions of firmware or software.

Finally, we assume that a subsystem that has no reported attack vectors is more secure than one that does. This work does not focus on zero-days because there is currently no way---at the design phase---to assess zero-day vulnerability just by analyzing the architecture of the system. However, our approach also works using private or proprietary vulnerability databases. As long as there exist historic data on the given subsystem, this approach allows an analyst to identify them and better inform system designers. This assumption is borne out in the literature. In attempting to violate the security posture of a system, a given intelligent threat actor will, most likely, use a set of techniques they are familiar with rather than come up with new techniques tailored to the individual system [Allodi, L., Massacci, F., & Williams, J.].

[7] CVE-SEARCH PROJECT, cve-search.org (perma.cc/5J2M-VAGC)

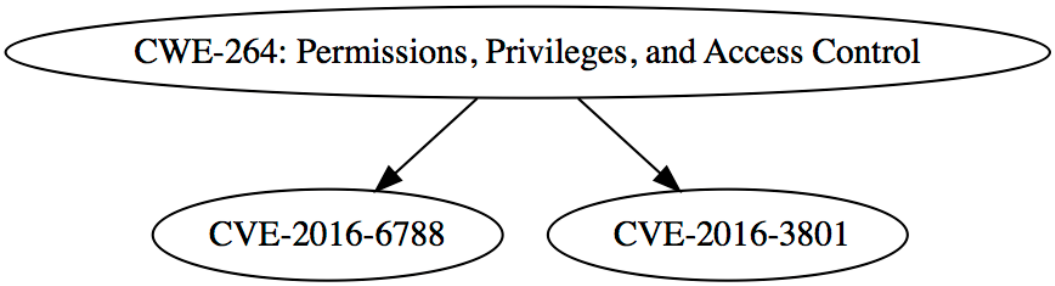


Figure 8 The two possible Adafruit Ultimate GPS attack vectors found in CVE share the same CWE category. This can inform the designer about general issues with the specific, chosen subsystem and to possibly look at substitutes with no reported vulnerabilities. Otherwise, the designers might choose to clearly document this class of vulnerabilities and construct solid requirements to mitigate such possible violation of system assets.

RESULTS OF MODELING TOOLS

This section describes a security assessment of two components in the FCS that can cause full degradation of expected service through exploiting historic vulnerabilities. Specifically, this analysis focuses on (1) the NMEA GPS device, which is necessary to provide location data and (2) the radio module, which is necessary to communicate with the ground control station or in the instance of manual control, the operator.

NMEA GPS Given the specification of the Adafruit Ultimate GPS (Table~\ref{tab:attributes} and Figure 6) we search through CVE to find two possible attack vectors (Figure 8). The first is CVE-2016-3801, which is a reported vulnerability specific to Android devices and targets the Mediatek GPS driver in the embedded operating system by crafting an application that can exploit the driver to \textit{gain} system privileges. Even if without an Android device, it is possible to misconfigure or program the FCS in such a way that the attack vectors are applicable, making this a threat one must account for when designing the system. The second is CVE-2016-6788, an attack that targets MediaTek I2C drivers and subsequently allows an attacker to \textit{elevate} their privileges and execute arbitrary code. This vulnerability was also reported for Android but, again, it can be applicable to the design of the FCS. Whilst both attacks take advantage of the GPS, they actually target the primary application processor (Figure 4.5) with possibly devastating effects to the system's expected service.

Further, the two vulnerabilities can form an attack chain by first using CVE-2016-3801 to gain access to the system, which would require an operator possibly accepting a request from the attacker, and then using CVE-2016-6788 to further elevate their privileges without having to go through operator input. It would be difficult to find these attacks without decomposing the

NMEA GPS device down to its specific implementation including the chipset it is employing and the required firmware to operate. Hence, system designers would have been unaware of this possible attack chain and would have to add further security considerations for this component post-deployment, instead of switching it with another that has no historic reported vulnerabilities and, therefore, presumably less risk.

Radio Module Another part of the system that could be attacked is the XBee radio module, which requires drivers for the ZigBee IEEE 802.15.4 protocol. Its specification can be seen in the SysML diagram (Figure 6). One of the possible attacks is described in CVE-2015-8732 and CVE-2015-6244 (these attacks have different bugs associated with them but result in the same effect), where an intelligent threat actor constructs packets that cause out-of-bound read and application crashes, resulting in a successful denial of service. Without radio communication the FCS would not be able to coordinate with the ground control station and it would go to a fail-safe mode, which could be detrimental to the mission it was planned to carry out. Knowledge of such attacks to the system's designers can lead to choosing a more robust, security wise, radio module for the implementation of the FCS.

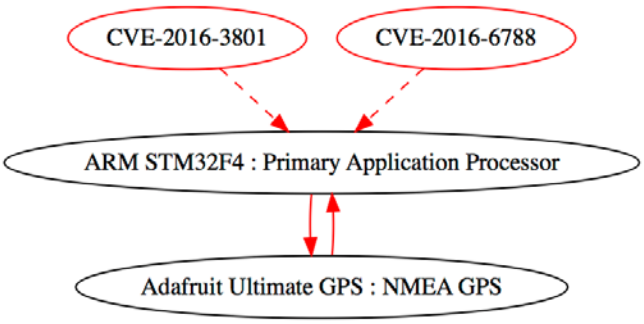


Figure 9 An intelligent threat actor can potentially take advantage of the use of the Adafruit Ultimate GPS drivers and can completely violate the systems expected service by escalating their privileges by either using the attack vectors presented individually or for a higher impact in sequence (attack chain). The dashed red edges indicate a given attack step, while the red solid edges indicate violation of data exchange. Since one of the attacks can lead to arbitrary code execution it can violate any path from the microcontroller to the sensory systems, meaning that such an attack would cause full degradation of expected service.

DISCUSSION OF MODELING TOOLS

The aforementioned example results further illustrate the need for early detection of vulnerabilities. Whilst this work has not evaluated elements of the attack surface--that is, components exposed to intelligent threat actors that can act as entry points into the systems structure--this paper demonstrates that a model can assist in generating security by design. This approach uses evidence (the historic data from open databases) to design the systems with

hardware and software that is historically more secure at no additional cost except those costs required for the security analysis. If architectural changes are not an option, it is still crucial to be aware of possible vulnerabilities and impose clear system requirements to preempt or mitigate against classes of attacks.

Without the generalized taxonomic scheme and its generated specification for the system, we would not have been able to match evidence to subsystems. This could have led to insecure systems getting deployed for safety-critical applications, which in turn can cause hazardous behaviors or, in the worst-case scenario, controlled accidents by intelligent threat actors.

In this section we have presented a framework that derives a characteristic set of attributes for each given subsystem in a CPS. These attributes construct well-formed models that are sufficiently detailed to allow for security posture evaluation of the system they specify. This framework is built on the examination of historic vulnerability data from databases, termed evidence, which apply to the system model based on those attributes. We have shown that this framework produces model sufficiency by mapping attack vectors for a possible NMEA GPS and radio module. While the method is agnostic to the modeling language, we represent the system in SysML, which is ubiquitous in systems engineering.

A future direction can use the findings of this research to automate the process of matching attack vectors. Towards this automation, we have shown the versatility of the method by transforming the SysML model to a generic graph metamodel using a standard graph schema based on XML, GraphML. A possible extension in this direction is to use the extracted system structure and apply techniques from computational linguistics to automatically produce attack vectors for each subsystems. This may allow cyber analysts to assess the security posture of increasingly complex systems.

CYBER ANALYST DASHBOARD

The Cyber Analyst Dashboard is a unified view between the mission specification model (Figure 11) and its associated attack vector space (Figure 10). The mission specification includes mission requirements as output from the War Room, admissible behaviors captured from the STAMP analysis and further refined in the War Room, and a potential system architecture that implements the defined mission. The attack vector space is all attack vectors from publicly accessible vulnerability libraries (e.g., CAPEC, CWE, and CVE), termed evidence. Ultimately, the individual attack vectors that are relevant to the system architecture model the potential attacker actions that they can take on the deployed system. This allows the analyst to do an initial, but supported, prognosis of the threat level. Through that prognosis the stakeholders can then take remedial action (in the form of architectural changes or resiliency) to protect the system from the threats it is modeled to face.

The functions provided by the User Interface (UI) should allow the analyst to navigate the mission-critical subsystems, visualize how these subsystems provide the admissible behaviors

and how they relate to the mission-level requirements. Additionally, the analyst will be able to navigate evidence that violates these subsystems as well as trace violated admissible behaviors and requirements in the presence of a reported attack vector.

Specifically by using the Cyber Analyst Dashboard, the cyber analyst should have a comparable view to the systems attack vector space, including potential attack patterns, system weaknesses, and recorded vulnerabilities for the specific software run on the system. By merging these views, the cyber analyst should be able to inform himself about the security posture of the system holistically and either find architectural preemption or mitigation techniques to address the attack vectors he deems important. This way he can better inform and communicate with system designers that then can provide architectural preemption and mitigation strategies or construct concrete system requirements to address these issues. Indeed, through this tool the cyber analyst should be able to communicate his findings to other stakeholders, for example system designers and operators, by presenting a common view through the model and its associated attack vector space.

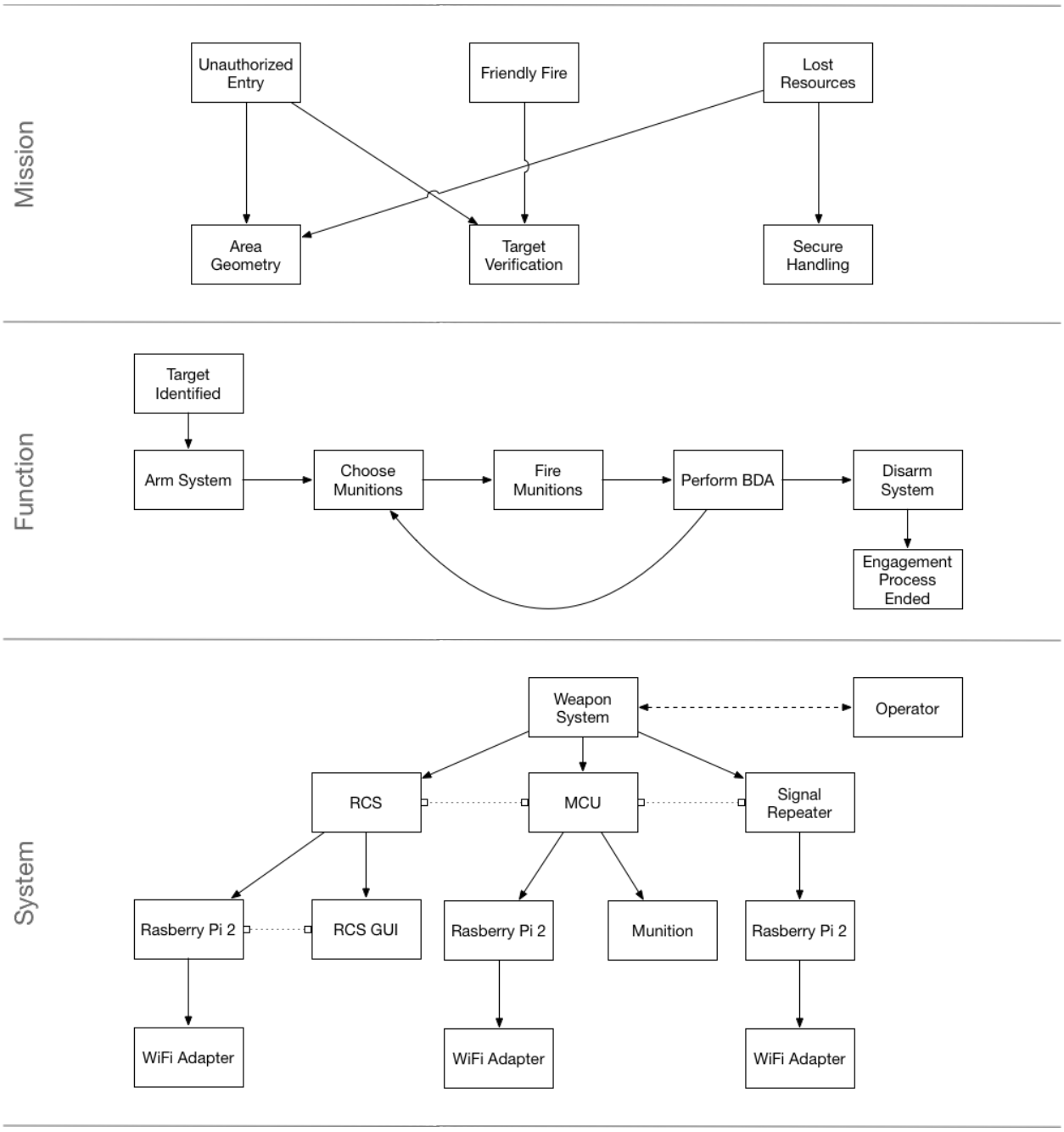


Figure 10 Mission, Functional, and System domains as a wireframe of how they will be presented in the Cyber Analyst Dashboard.

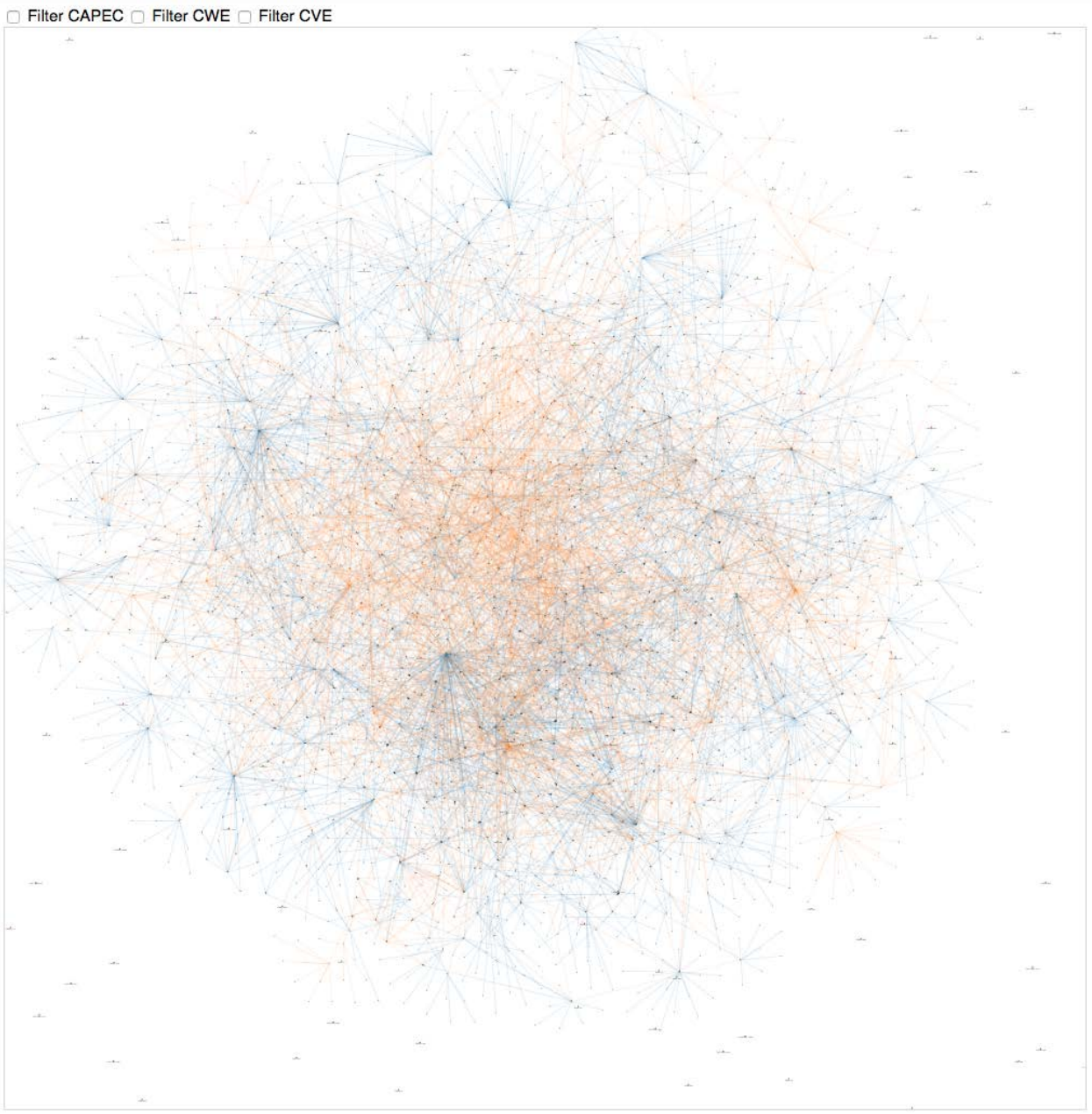


Figure 11 ***Attack vector graph for CAPEC, CWE, and CVE. This shows the filtering functions. The graph also includes interactivity functions of the form of tooltips to show further information, zooming in or out, or moving nodes to show clusters of attack vectors as defined by their intra and inter relationships.***

FEATURES

The major features we will implement in the Cyber Analyst Dashboard are:

1. Import functions for GraphML files.
2. Visualization functions for the associated graphs.

3. Interactivity functions for manipulating the graph and presenting different visualization information to the cyber analyst.
4. Search functions to refine the graphs to deal with their intrinsic complexity.
5. Addition function to add cyber analyst specific information, e.g., adding an attack vector to the attack vector space graph that might not be included in the databases but the analyst knows it to exist within the system structure and associate it to a specific architectural element.
6. Trace functions for the mission specification graph.

IMPLEMENTATION

The current preliminary (pre-alpha) implementation utilizes the D3.js library. The Cyber Analyst Dashboard can import a prespecified GraphML file (either of the system structure or attack vector space) and use a force-directed graph layout to visualize the information contained within them. It can also filter based on database. For example in the attack vector graph that contains all of CAPEC, CWE, CVE one can choose which of those libraries to show instead of having to work with the full attack vector space that can be hard to navigate. Once CYBOK is more mature, we will be able to filter based on component, a series of components, or any other behaviors and requirements based on the output from the tool.

For the beta implementation of the tool, several libraries can provide a higher degree of versatility in implementation:

1. D3.js (JavaScript) & cola.js (JavaScript)
2. cytoscape.js (JavaScript)
3. plotly (Python)

From the three the most seasoned is the first, D3.js, which offers a force-directed graph layout. An extension of D3.js, cola.js, allows for constrained force-directed layout, which could allow an automated visualization of the three domains in the mission specification. The second, cytoscape.js, provides a wide selection of graph drawing algorithms. This follows naturally since cytoscape (as both standalone software and JavaScript API) geared only towards graph structures. Having said that, cytoscape.js is a new, rapidly changing API (albeit somewhat more stable than a year ago when this question first arose). Finally, the newest and least seasoned API is provided by plotly. This library is also restricted to only certain graph drawing algorithms, since its purpose is to be more general and more idiomatic compared to the above options.

The use of more than one API is possible but not recommended since the libraries can have the same dependencies but not necessarily relying on the same version (this is mostly true for the first two libraries that are based on JavaScript, integrating a python library with a JavaScript library would be a task in and of itself).

Another possible developmental tool to consider is react.js. A library that allows a user to programmatically modify DOM elements (which all three above libraries can above can do) and provides smoother interactivity than either of the visualization libraries detailed above. React.js can be used in conjunction with any of the libraries defined above. Another similar approach is to use the Electron framework to create a standalone application that is operating system agnostic.

DOCUMENTATION

Programming language (that is JavaScript) documentation:

- JavaScript reference https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Developmental tool documentation:

- d3.js documentation https://github.com/d3/d3/wiki
- cytoscape.js http://js.cytoscape.org/
- plotly API reference https://plot.ly/api/

Graph format (that is GraphML) documentation:

- The GraphML file format http://graphml.graphdrawing.org/
- GraphML format (from Gephi) https://gephi.org/users/supported-graph-formats/graphml-format/

CYBER BODY OF KNOWLEDGE (CYBOK)

What is CYBOK?

The goal of this task was to develop a Cyber Body of Knowledge (CYBOK) Information Retrieval tool for systems engineers, cyber analysts, and requirements engineers working in the Mission Aware Framework to discover relevant attack information at the earliest possible stage of systems development. CYBOK is model driven and data-driven so that cyber information (attack patterns, vulnerabilities, and weaknesses) most relevant to system model can be used to drive system vulnerability detection and mission impact analysis in the Mission Aware framework.

WHERE CYBOK FITS INTO THE MISSIONAWARE FRAMEWORK

Referring to Figure 12 below, MissionAware first defines the possible mission scenarios and then it identifies both the possible mission hazards but also the type of threat space that is potentially going to be associated with the system architecture. The WAR ROOM is the fundamental concept in MissionAware. The WAR ROOM produces a body of information that drives system hazard analysis and SysML modeling efforts – that capture the mission requirements, admissible behaviors of the system, architectural features of the system, identification of Hazards, identification of critical assets, and assessment of high level cyber threats. The SysML modeling activity takes WAR ROOM outputs and encodes mission critical cyber information into workflow

models to understand the potential threat space associated with the mission. CYBOK is used to help the system analyst gauge the relevant vulnerabilities (and associated attacks) of the system and threat actors. Through the CYBOK databases and cyber analyst dashboard, the analyst extracts vulnerability information that is potentially applicable to the mission and the system architecture. CYBOK's Information Retrieval process does the preliminary steps to this by using the system model to identify relevant attack patterns, weaknesses, and vulnerabilities. As shown in Detailed Architecture sub-section of this section, the results are evaluated at various levels of granularity, and in multiple compositions, to assess whether relationships between model elements align with common interfaces between attacks, which should give insight into whether an attack chain may be composed along a path in the system.

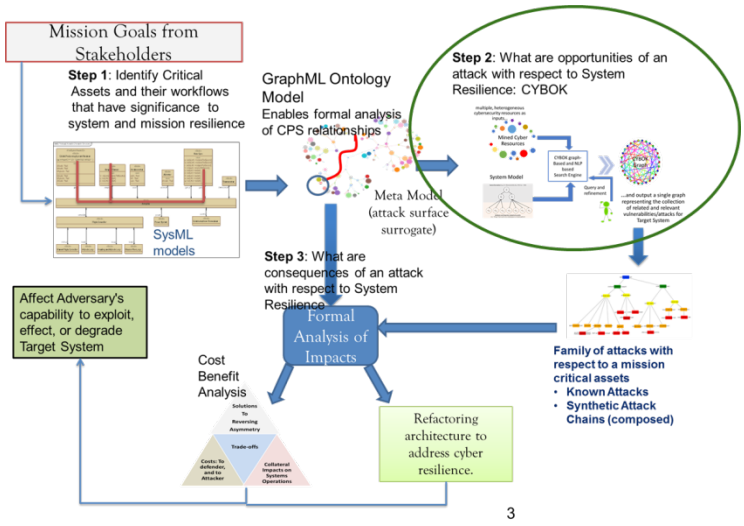


Figure 12 ***Where CYBOK Fits into MissionAware***

RATIONALE FOR CYBOK

Cybersecurity design requires taking the attacker's perspective to best understand how to defend a system from threats [Kordy, B., Piètre-Cambacédès, L., & Schweitzer, P. (2014)]. However, this attacker's perspective requires expert domain knowledge and it is time-consuming to express these requirements in a sufficiently general way for systems engineers to understand. Recently, Security Design Patterns and Attack Patterns have emerged as means to organize the design of cyber-secure systems around "vetted and reusable" solutions that are more comprehensible. The Common Attack Patten Enumeration Classification (CAPEC) database by MITRE is one example of well-formed Attack Pattern classification schema. Unfortunately, Attack Patterns, as defined there, are infrequently used in the systems requirements and design phase, primarily because of (1) existing "perimeter or after-the-fact security solutions" dominate the IT landscape, and (2) the *applicability* of Attack Patterns is still difficult for system engineers to use in the systems framework.

The aim of CYBOK is the creation of a tool which curates cyber security domain knowledge (e.g., CAPEC, CWE, CVE), to provide usable information to both the security analysts and system engineers. If multiple cyber threat repositories can be used in congress to create a "Unified Cyber Body of Knowledge" then a synergistic effect is realized - used to design and assess systems that are more cyber resilient. How to achieve and realize a "Cyber Body of Knowledge" in a system based engineering context has been one of the key challenges for this sub-task. A related challenge has been developing engineer accessible automation tools for "Cyber Body of Knowledge" to allow system engineers to search over a large attack pattern space, create new attack patterns to aid system vulnerability analysis and mitigation techniques. In 2017 we made breakthrough progress with respect to these two challenges.

Our prior efforts were mainly concentrated on creating tools to assist analysts on finding suitable attacks from attack libraries (like CAPEC). While these tools were necessary first step along the path, they did not achieve the full vision of a Cyber Body of Knowledge from a Systems Engineering POV.

***Key aspects of CYBOK approach**:*

1. CYBOK is a multi-view search engine on how to "relate" cyber threat information in a systems model context. It views the diverse set of cyber repositories (CAPEC, CWE, CVE, CPE, etc.) as greater than the sum of their individual parts. Uncovering the synergistic relations in these diverse set of repositories and casting the information into "system" model perspective is the innovative aspect of CYBOK.
2. CYBOK uses weighted and unweighted query terms and retrieves all entries, initially ranking them by similarity to the query.
3. CYBOK generates a set of queries from a graph model of the system, creates aggregate summaries of the search results, and creates a direct association between components and attacks for further analysis.
4. CYBOK is ontology based; it uses data science methods, Natural language Processing and graph methods to mine cyber-attack data and "connect it" to SysML models of the system to drive cyber vulnerability detection and analysis of the system.
5. CYBOK Information Retrieval is *driven* by the system perspective – a SysML model, mission requirements, and cyber assurance needs. Information from the SysML model of the system is distilled into a graph schema (GraphML) – automatically. The compact GraphML model carries the necessary model information for CYBOK to search. CYBOK produces as output attack patterns and associated vulnerabilities with respect to the system model.
6. The results obtained by CYBOK can be easily examined, iteratively modified and decomposed and disseminated among the developers.

OVERVIEW OF CYBOK

CYBOK uses multiple databases which each provide a unique perspective on the offensive and defensive views of a system, and on a range of abstraction levels. This creates the foundation for the CYBOK concept – that unifying these views and learning from the many interrelationships among the data in CYBOK and a system-of-interest will aid in successfully identifying weaknesses in mission critical systems and guide us toward strategic cyber defense principles.

As depicted in Figure 2, CYBOK takes as input a SysML derived model of the target system. Specifically, CYBOK takes the GraphML representation of the SysML model. GraphML meta model is encoded with specific cyber attributes that define relevant attack surface of functions and components [Bakirtzis, G., Carter, B. T., Elks, C. R., & Fleming, C. H. (2018)]. The GraphML meta-model is automatically generated from SysML by our tools. The importance of the GraphML model is that it encapsulates the important model relations, properties, functions, and attributes in the SysML model in a compact formalism. With this input, the CYBOK Search Engine generates queries from the model and builds the initial search results from matches between the text and the inverted index. These results are ranked according to their relevance, as computed using traditional NLP methods (e.g. Cosine Similarity and TF-IDF). Within a set of results, we may view this subset (or subgraph) of CYBOK as a 'family' of historical and ontological knowledge concerning the query. The CYBOK Construction Engine's Data Mining efforts aim to identify the key interrelationships between and among these families, such as consistent severity scores or common mitigation strategies; the information gained from these efforts may be used to re-order the results according to some user preference to drive Vulnerability Detection and mitigation.

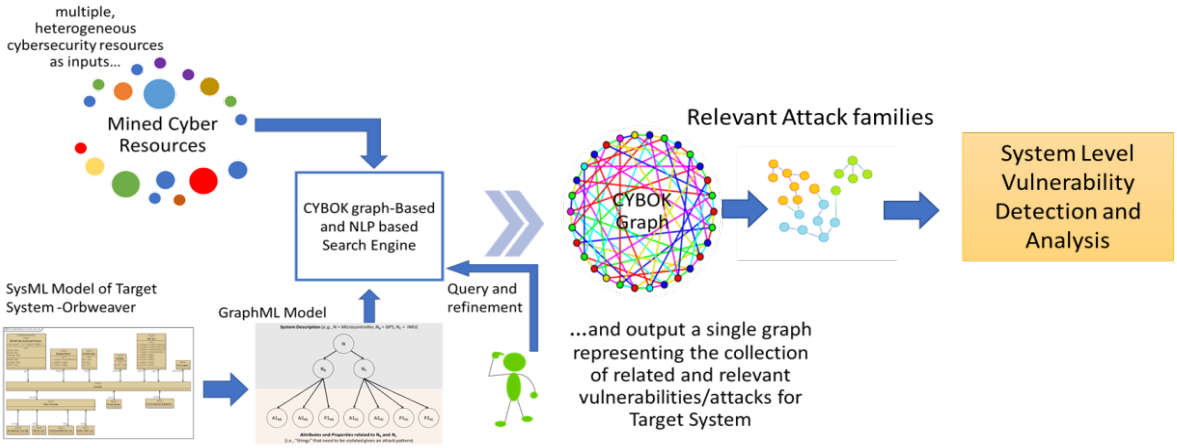


Figure 13 ***Conceptual Overview of CYBOK - the key innovations of CYBOK are; (1) search functionality in relation to models of the system – not just searching CAPEC/CWE/CVA for cyber-attack patterns/vulnerabilities; (2) using NLP and data analytics to infer new clusters of vulnerabilities and attack patterns so that a more complete picture of the threat space can be applied to the system.***

At present we have integrated four databases/catalogs (CAPEC, CWE, CVE, CPE). See Table 5 below. Each of these databases serve different roles in Cyber Analysis. Some inform about attacks patterns, some on common weaknesses, others tabulate known vulnerabilities in platforms and software. As we can see from Table 5, these databases have diverse focal points within the cybersecurity domain, and they may have direct relationships to one another or relate to similar information concepts, e.g., attacks, platforms, etc.

Resources such as CAPEC may be explored on the web, albeit in restricted manner as they must be searched by ID's or simple natural language queries – which is inefficient for more complex systems. It is often the case that a single search query will insufficiently describe the necessary semantic qualities of the system. Finally, while the existing resources describing attacks, weaknesses, and vulnerabilities each present their own perspective, as in CAPEC, CWE, and CVE databases, respectively, none is sufficient on its own, and thus a reliable information retrieval tool integrating these data collections is needed. This need is further compounded by the fact that these collections share meaningful references to one another, such as a weakness being targeted by an attack pattern. CYBOK is based on an ontology search framework that allows the synergistic relationships between entries in the databases to be exploited to the maximum extent.

Table 5 Examples of Cybersecurity Resources Used in CYBOK

Cybersecurity Resource	Focus	Representation	Size	Known relationships	Data format
CAPEC (Common Attack Pattern Enumeration Catalog	Pseudo-ontology of Attack Patterns	Hierarchical Graph	510 Attack Patterns	Links to CWE, CVE	HR* Text, common tech. words
CWE (Common Weakness Enumeration)	Pseudo-ontology of Weaknesses and Vulnerabilities	Hierarchical Graph	714 Vulnerabilities, Exposures, or Weaknesses	Links to CAPEC, CVE	HR* Text, common tech. words
CPE (Common Platform Enumeration)	Provide universal identifiers for software platform (single or multiple versions), as requested by Vendors	Instance-based	123,324+	Used by CVE	Specially formatted; see the CPE specification for details. Uses platform specific names
CVE (Common Vulnerabilities and Exposures)	Repository where Vendors may announce vulnerabilities found in their software	Instance-based	93,546+	Vendors using CPE, may use the CPE Name for the affected software version(s); Links to CWE	Brief HR* Text, with additional info such as CVSS scores
Exploit DB	Code repository for PoC cyber-attacks	Organized by Target Platform	38,242+		Program code, languages vary; some HR* Text

US-CERT (U.S. Computer Emergency Readiness Team	Announcements of Cyber threats to be aware of	Instance-based, Temporal	??		HR* Text; with additional information

HR* - Human Readable

There are numerous resources in the cybersecurity domain, not limited to CAPEC, CWE, CVE, and CPE. We have selected these four to incorporate into CYBOK because they either a) comprehensively describe important concepts of the attacker or defender's perspective, or b) explicitly relate platforms to known vulnerabilities.

CAPEC, is a pseudo-ontological hierarchy of cyberattacks. It describes attacks based on techniques used to accomplish them, as well as with respect to the goal of the attack (such as collecting information or manipulating a state). There are over 500 attack patterns in it, described in natural language, with content ranging from very concise descriptions to attack execution flows to detection and mitigation strategies. A deficiency of this collection, beyond its incomplete entries, is that it rests at a high level; even low-level attack patterns are rarely specific about applicable languages or platforms. However, numerous attack patterns refer to weaknesses in CWE (Common Weakness Enumeration) that they target, and few refer to CVE (Common Vulnerability Enumeration) instances of platform vulnerable to such attacks.

CWE weaknesses are organized according to multiple views, such as where in development the fault arises, or by abstractions of the software behaviors. CVE is a repository where vendors may report the presence and status of an exploitable vulnerability in an affected platform. Descriptions are short and do not always state the applicable attack. These instances may also contain CVSS scores (a widely accepted scoring system for vulnerabilities) and references to CWE. Finally, each CVE instance possesses a list of all CPE identifiers for affected versions of the platform.

CPE (Common Platform Enumeration) is a unified naming convention for software and hardware platforms, where vendors may catalog versions of their platforms. Each of these resources provides a unique perspective on the problem of security, whether it is from the attacker or the defender's side. While this describes the resources integrated within CYBOK at this point, we consider these data sources preliminary, and plan inclusion of other datasets in the future.

Figure 14 shows the high level relationship between the repositories. Datasets composing CYBOK (such as CAPEC, CWE, CPE and CVE) each present a different perspective. Some are attacker oriented, and may have a range of specificity about the target system or platform. While others are Defender/System owner -oriented. While CAPEC and CWE are in general, high-level there is *relative* variance in platform specificity as you go through the hierarchies. The perspective and specificity features of the databases when used in a "search engine context" provide the analyst with richly featured ecosystem to gather cyber data about the system.

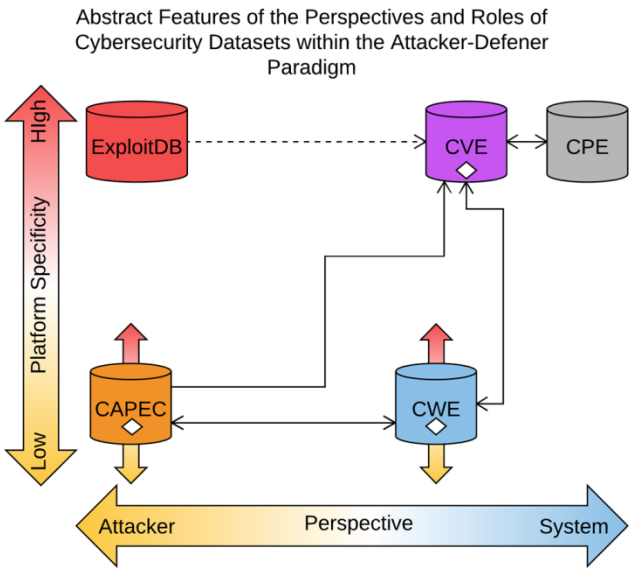


Figure 14 ***Different Perspectives of the Cyber Databases***

MAJOR PRODUCTS

Addressing the first major task of CYBOK, we implemented an efficient information retrieval tool which can decompose a query string into parts (n-grams) and build the initial text-based search results at little cost – the major workload for computing the relationships between terms and documents in the collection at large will have been done in the construction phase.

- This tool uses an inverse index mapping from terms, or rather n-grams, to documents, and precomputes the normalized TF-IDF of each term for each document the term occurs in. At the point of query execution, query dependent weights are calculated and used to compute a weighted sum of partial results (dictated by the n-grams in the query string).
- TF-IDF has been a dominant method in NLP, and more specifically for information retrieval, for its effective ability to discriminate commonality of words at a local and collection-wide scale. In this work, an ongoing effort concerns the relationship between the levels of a hierarchical system of documents, as opposed to an unordered collection, and how aspects of TF-IDF may be extended to better suit the hierarchical setting.

The second major task, we implemented a low level, and two higher level methods of performing the searches in CYBOK.

- Most similar to search engines like Google, the first of the high-level methods takes a string of natural language text from the user, preprocesses it, followed by iterative construction of the results from the matching indices of the inverted index.
- The second method takes an XML document and performs a search over a query group.
  - Specifically, multiple queries designated as being related may be done in unison, such that all terms in each individual subquery are weighted according to the overall query group. That is, the results for each subquery are dependent.

- Our next steps involve assessing how to merge partial results computed for one query with those of another query, and how to evaluate that against the explicit and implicit relationships among the groups. This task is tightly related to that of custom rankers, where candidate methods of merging results involve user-driven ordering of attributes for the purpose of defining an attribute-based scoring metric.

In the low-level method, the user directly assigns the weights to each query term. This may be useful for performing minor iterative modifications to a preprocessed query to clean up certain elements from the associated results, such as by decreasing the weight of an over-represented term in the search results.

TOPIC MODELING THE CAPEC LIBRARY

Topic modeling [Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003)] is a machine learning technique commonly used in natural language processing (NLP) that estimates latent or hidden topics from a corpus of documents. Often in NLP, documents are described by a vector of the number of times each word appears in the text. We will represent the length of this bag-of-words vector using L. For large documents and corpuses, this vector can quickly grow which leads to numerous problems when attempting perform analytics. Topic modeling estimates a topic distribution that can be used to describe each document in the corpus. Let the number of topics will be represented by T. The topic distributions, where generally $T \ll L$, can be used in place of the bag-of-word vectors when performing data analytics. Further, topic modeling can also uncover new relationships between documents that the bag-of-words method does not represent.

In this project, we investigated the use of topic modeling on the CAPEC. CAPEC is composed of 512 common attack patterns. Each attack pattern was a textual description ranging over many fields. Common fields include a summary of the attack, attack prerequisites, and links to related attack patterns.

We believe that topic modeling can be a useful tool for automatically extracting information about each attack pattern from the CAPEC library. We envision two primary roles for topic modeling the CAPEC library in the broader scope of this project. First, the estimated topic distributions can be used to cluster the attack patterns and extract new information about the relationships between attacks not provided by the CAPEC hierarchical structure. Second, topic modeling can provide a method for linking attack patterns to models of the physical system we wish to protect. This is achieved by estimating a topic distribution of the text in the model of the system and then finding the attack pattern with a similar topic distribution. Distance between the attack topic distribution and the model topic distribution can be measured using standard methods for comparing probability distributions such as the Kullback-Leibler divergence [Kullback, S., & Leibler, R. A. (1951)].

In 2017, work involving topic modeling and CAPEC focused on testing topic modeling algorithms and seeing if the algorithm can extract some form of new information from the library. Future

work will further refine and improve this initiative. Only preliminary work has been performed on the second role of topic modeling with the bulk of this effort being left to future work.

In this section, we will first describe the mathematics behind topic modeling. There are several types of topic models including hierarchical topic models [Griffiths, T. L., Jordan, M. I., Tenenbaum, J. B., & Blei, D. M. (2004)], correlated topic models [Lafferty, J. D., & Blei, D. M. (2006)], and supervised topic models [Mcauliffe, J. D., & Blei, D. M. (2008).], but we limit our discussion to the basic latent Dirichlet allocation (LDA). We then present our analysis of topic modeling on the CAPEC library.

TOPIC MODELING

LDA assumes that each document in a corpus is represented by a mixture of random topics and that each topic is represented by a distribution over words. In the basic version of LDA, the presence of a word in a document is used instead of the word count. Let v be a vector of binary variables that represent the presence of words in a document where if $v_i = 1$ then the i^{th} word appears in the document and if $v_i = 0$ the i^{th} word does not appear in the document. A particular document is composed of a sequence of words with length N and is denoted by $w = (w_1, \dots, w_N)$. The corpus is composed of M documents and represented by $D = \{w_1, \dots, w_M\}$.

When using LDA, the following generative process is assumed for each document in the corpus:

1. Randomly sample N from a Poisson distribution with rate parameter ξ
2. Randomly sample θ from a Dirichlet distribution with parameter α
3. For each word in N:
    a. Randomly sample a topic z_n from a multinomial distribution with parameter θ
    b. Randomly sample a word from a multinomial distribution dependent upon the topic $P(w_n|z_n, \beta)$

In this process for generating each document, it is assumed that the number of topics T is known and fixed. The joint distribution of the topic mixture θ, the set of topics z, and the set of words w given the parameters α and β is given by:

$$P(\alpha, \beta) = P(\alpha) \prod_{n=1}^{N} P(\theta) P(z_n, \beta).$$

The marginal distribution for the set of words can be found by integrating over θ and summing over the topics:

$$P(\mathbf{w}|\alpha, \beta) = \int P(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{\mathbf{z}} P(z_n|\theta) P(w_n|z_n, \beta) \right) d\theta.$$

The marginal distribution of the corpus can be found by multiplying the marginal probabilities of each documents:

$$P(\mathcal{D}|\alpha,\beta) = \prod_{d=1}^{M} \int P(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{\mathbf{z}_d} P(z_{dn}|\theta) P(w_{dn}|z_{dn},\beta) \right) d\theta_d.$$

Figure 15 displays a probabilistic graphical model of the LDA.

The key problem for LDA is estimating the hidden topic distribution z and the parameter θ given a document. The posterior for these two variables is given by:

$$P(w,\alpha,\beta) = \frac{P(\alpha,\beta)}{P(w|\alpha,\beta)},$$

where

$$P(\mathbf{w}|\alpha,\beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{L} (\alpha_i \beta_{ij})^{w_n^j} \right) d\theta.$$

The posterior distribution is intractable for an exact solution but other estimation methods, such as variational inference, can be employed to estimate the hidden variables.

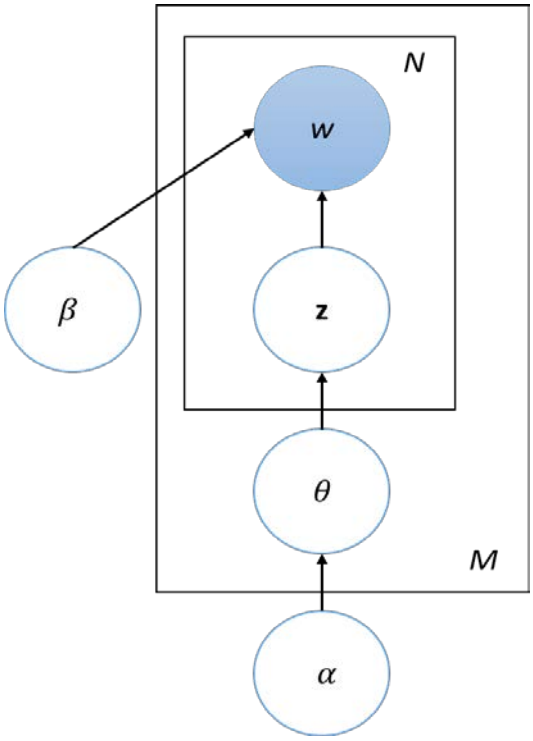
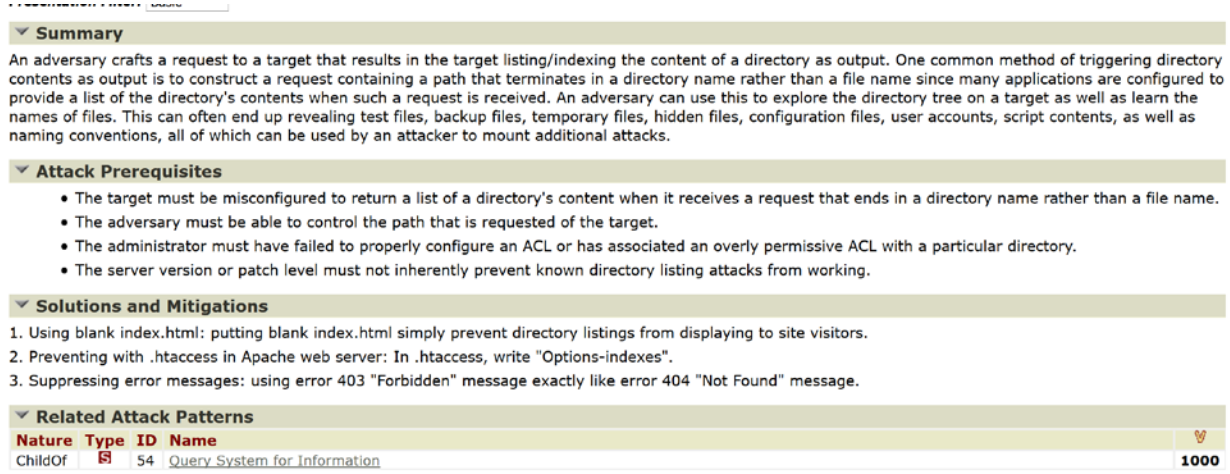


Figure 15 ***Graphical model of latent Dirichlet allocation***

CAPEC ANALYSIS

CAPEC consists of written descriptions of numerous attack patterns. Figure 16 is a screenshot of an example attack pattern from CAPEC. The content for each attack pattern varies greatly. Some attack patterns do not contain any text while others have extensive sub-topics and detail.



Summary

An adversary crafts a request to a target that results in the target listing/indexing the content of a directory as output. One common method of triggering directory contents as output is to construct a request containing a path that terminates in a directory name rather than a file name since many applications are configured to provide a list of the directory's contents when such a request is received. An adversary can use this to explore the directory tree on a target as well as learn the names of files. This can often end up revealing test files, backup files, temporary files, hidden files, configuration files, user accounts, script contents, as well as naming conventions, all of which can be used by an attacker to mount additional attacks.

Attack Prerequisites

- The target must be misconfigured to return a list of a directory's content when it receives a request that ends in a directory name rather than a file name.
- The adversary must be able to control the path that is requested of the target.
- The administrator must have failed to properly configure an ACL or has associated an overly permissive ACL with a particular directory.
- The server version or patch level must not inherently prevent known directory listing attacks from working.

Solutions and Mitigations

1. Using blank index.html: putting blank index.html simply prevent directory listings from displaying to site visitors.
2. Preventing with .htaccess in Apache web server: In .htaccess, write "Options-indexes".
3. Suppressing error messages: using error 403 "Forbidden" message exactly like error 404 "Not Found" message.

Related Attack Patterns

Nature	Type	ID	Name	V
ChildOf	S	54	Query System for Information	1000

Figure 16 ***Screenshot of example attack pattern from CAPEC***

The attack patterns are hierarchical organized based on domain knowledge. There are nine top categories:

1. Collect and Analyze Data
2. Inject Unexpected Items
3. Engage in Deceptive Interactions
4. Manipulate Timing and State
5. Abuse Existing Functionality
6. Employ Probabilistic Techniques
7. Subvert Access Control
8. Manipulate Data Structures
9. Manipulate System Resources

A small number of the attack patterns do not belong to one of these categories. There are numerous sub-categories and the depth of the hierarchy can vary. Each attack pattern is assigned to a node in the hierarchy.

All information in the CAPEC library can be downloaded to an XML file. The text of each attack pattern and the associated category is extracted from the XML file. The data extraction and all other analysis in the section was performed using the R statistical software. The *XML* package for R was used for manipulating the XML file.

In our first experiment, we try to predict the category using the counts of each word. The purpose of this experiment is to test for correlation between the text and the category and to give us a

baseline for evaluating topic modeling. Logistic regression has been widely used for this type of analysis, and the *LiblineaR* package in R was used to test the accuracy of predicting the label using this model. Using a 10-fold cross validation procedure, the accuracy of predicting the category using the word counts was roughly 72%.

A topic model with 50 topics is estimated using the *topicmodels* R package. The estimated topic distributions are used as inputs into the logistic regression model, and we test the predictive ability of the topic distributions on the category as above. However, the 10-fold cross validation accuracy falls to 56%. Further, we randomly divide the attack patterns into testing and training sets and build a random forest classifier using the *randomForest* package on the training set. We predict the categories of the attack patterns in the testing set and the accuracy falls to 45%. This analysis demonstrates that word counts are better at predicting the assigned CAPEC category than the estimated topic distributions. However, our objective is to extract new information about the attack patterns using topic modeling. In order to increase the interpretability of a topic model, we reduce the number of topics to 4 and estimate a new model. Based on the estimated distributions, we assign the following labels to each topic:

1. Mobile-based Attacks
2. Web/Client Side Attacks
3. Privilege Attacks
4. Data Corruption

When using topic modeling, distributions of topics are estimated for each document. This means that attacks are not assigned to a single topic but can have a mixture of topics. For example, the attack pattern "Using Alternative IP Address Encodings" is assigned a mixture of topics 2 and 3. Further, we found that if we give a hard assignment of a topic based on the topic with highest probability, there is a mixture of CAPEC categories in each topic. We conclude that topic modeling is extracting new information from the CAPEC library that can be used for new clustering methods beyond the assigned hierarchical structure.

FUTURE WORK WITH TOPIC MODELS

There are several aspects of topic modeling and the CAPEC attack patterns that still need to be explored. As previously mentioned, there are numerous types of topic models and this analysis is limited to the basic LDA method. Future methods could include supervision and correlation. Current topic modeling algorithms just use a bag-of-words approach which does not model the placement of the word in the document. The location of a particular word could be significant and we will develop topic models that take this into account.

Our other suspected use for topic modeling is to match attacks to systems. Our objective would be to select attacks for a particular system that have similar topic distributions. This would require estimating topic distributions for the system under study. We have performed some initial experiments estimating topic distributions using the text from a SysML model. However,

this needs further exploration. One advantage to this proposed method is that the topic model for the system can be estimated from numerous documents such as manuals.

DETAILED ARCHITECTURE OF CYBOK

Referring to Figure 17, The CYBOK architecture is composed of four major functions: (1) a Construction Engine, (2) a System Model Manager, (3) the CYBOK Search Engine, and (4) the User Interface. These handle the preprocessing of the databases to build the data model and successive data mining operations; the preprocessing, query generation, and exploration of the System model; query resolution and ranking of search results; and the management of user-controllable features, inputs, and parameters, in addition to presentation of results and statistics; respectively. The major parts work together to model and learn from the interrelationships between the instances of the collection, to allow the user to explore the information contained within CYBOK and their system, to generate queries either directly or indirectly from the system model, and to explore the implications of the search results with respect to the system or query. With this information, the aim is to give security analysts and developers alike the leverage to use this information in making secure design decisions.

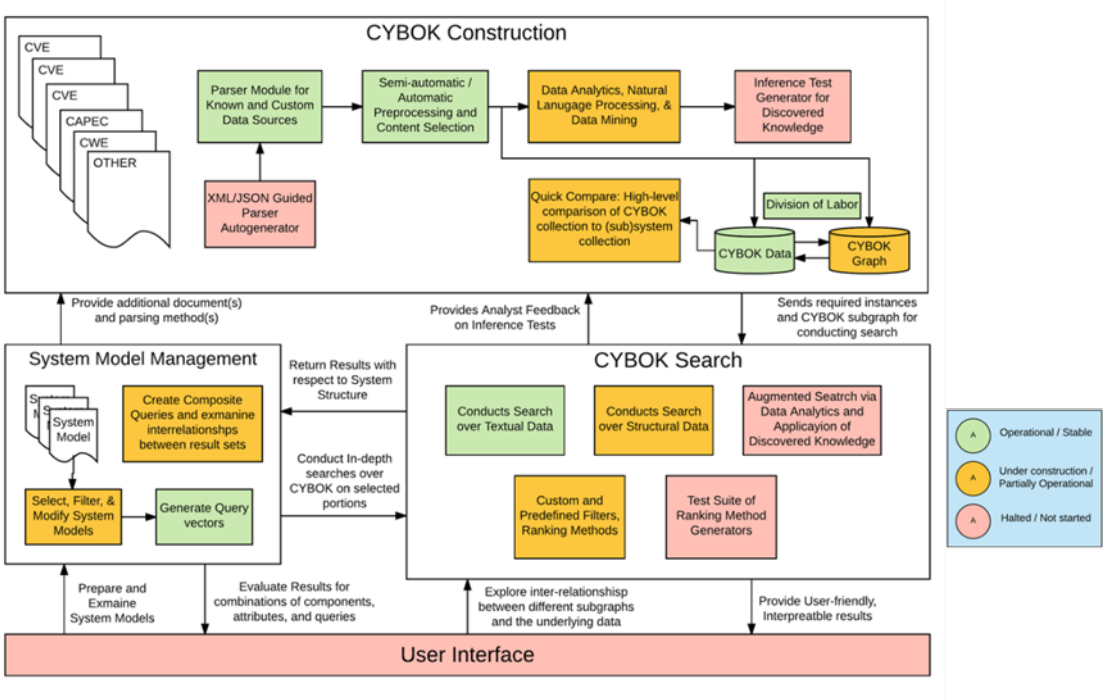


Figure 17 ***Architecture Implementation of CYBOK***

The CYBOK Construction Engine handles preprocessing and data model decisions concerning CYBOK, and the accompanying data mining operations on that data. Its main components are the preprocessing logic for parsing the documents and performing content selection, the

constructor for the inverted index which drives the CYBOK Search Engine, and the data mining and data handling operations.

- In preprocessing, the Construction Engine takes user-controllable parameters for the data representation, such as n-gram length and the files to be selected, and uses this to build CYBOK and the inverted index.
- Through the User Interface, the user may generate queries, explore, and filter the instances making up the CYBOK graph. That is, they may freely examine how information is interrelated irrespective of a target system.
- The existing data representation used at this stage models the documents according to the attributes and subject tags extracted from original source, in addition to the keys making up the inverted index which map to these instances.
- The data mining functions of the Construction Engine perform various data analytic methods upon results from one or more query and identify potential relationships between the queries and the instances comprising the results. Knowledge learned here may be cached for later use in evaluating related queries.

The System Model Manager handles user operations concerning the system model. Given a system model, it preprocesses it to get the data contained therein into a similar form to that of the information in CYBOK. The user may interact with the System Model Manager through the User Interface to explore the system model, to select subsystems of interest, and to generate queries from the model to pass into the CYBOK Search Engine. Query results may then be explored with respect to the system and assessed accordingly. Here, an analyst may also make iterative revisions to a query to see how it impacts the assessment.

The CYBOK Search Engine takes queries from the User Interface, the System Model Manager, and the Construction Engine to provide information about relevant attacks, weaknesses, and vulnerabilities to the provided query. It uses an inverted index to efficiently resolve text-based queries; the results may then be evaluated for their statistical, graph, and instance-derived properties. Further analysis must be done to assess viable mechanisms for ranking and filtering of search results according to the System model or query structure, and regarding the interrelationships between instances of the results.

The User Interface provides the interaction control between user and CYBOK. At initialization of a new CYBOK instance, the user may select which documents to include and the n-gram length. At a low level, the user may directly interact with the instance data and the inverted index to explore this information and to manually interface their search operations with analytic methods. From within the wrapper method for the search operations, the user may provide queries in a natural language or XML-based form, and have the results presented accordingly. Results at this level provide the ID of the instance, the name of the instance, and the score acquired for that instance in the search process. With this information, the analyst can perform some initial decision making about the results.

These major functions within CYBOK handle the processing, modeling, and learning from data gathered from the selected resources and the system model input by a user. They allow for a user to explore their system and CYBOK, gaining insight into the system context, the overall threat landscape, and how these are interrelated. The information returned from a search result may be more than necessary for the analyst (at this time), but it will be sufficient to cover the historical and ontological perspectives needed to evaluate the system.

HOW CYBOK WORKS

At startup, shown in Figure 18, the user selects the appropriate parameters for the CYBOK instance. First is the length of the n-grams, which should be a numeric value. Next, the user decides for each of CAPEC, CWE, and CVE whether to include them. After each, there is the option of selecting the file if the user entered 'y'. The initialization function (e.g. init_tool() in cybok_wrapper.py) returns a CYBOK class object, which can be used to search the index directly and examine the instances, or this can be passed into the search wrapper method (search_wrapper() in cybok_wrapper).

Once we have the CYBOK object and we call the search wrapper, we may choose thresholds, query methods, and enter queries. The leftmost column in Figure 6.8 denotes the ID's of the instances; these may be passed as parameters to the get_instance() method to get the object representing that document.

```
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cybok_wrapper
['nvdcve-2.0-2016.xml', 'nvdcve-2.0-2015.xml', 'nvdcve-2.0-2014.xml', 'nvdcve-2.0-2013.xml', 'nvdcve-2.0-2012.xml', 'nvdcve-2.0-
2011.xml', 'nvdcve-2.0-2010.xml', 'nvdcve-2.0-2009.xml', 'nvdcve-2.0-2008.xml', 'nvdcve-2.0-2007.xml', 'nvdcve-2.0-2006.xml', 'n
vdcve-2.0-2005.xml', 'nvdcve-2.0-2004.xml', 'nvdcve-2.0-2003.xml', 'nvdcve-2.0-2002.xml']
>>> cybok = cybok_wrapper.init_tool()
2017-12-04 16:51:16.835700
Enter ngram size (should be n>0; defaults to 2)1
import attacks (y/n)?y
"def" for default or file name:def
import weaknesses (y/n)?y
"def" for default or file name:def
import vulnerabilities (y/n)?y
"def" for default or "list" to enter one at a time:list
CVE file name: nvdcve-2.0-2016.xml
CVE file name: nvdcve-2.0-2015.xml
CVE file name:
Starting CAPEC:  2017-12-04 16:51:42.125563
Completed CAPEC:  2017-12-04 16:51:53.240584    #CAPECs:  518
Starting CWE:  2017-12-04 16:51:53.240635
Completed CWE:  2017-12-04 16:52:12.335825      #CWEs:  989
Starting CVE:  2017-12-04 16:52:12.335876
Completed CVE:  2017-12-04 16:54:10.806287      #CVEs:  16391
Indexing CAPEC:  2017-12-04 16:54:10.806344
Indexing CWE:  2017-12-04 16:54:15.505327
Indexing CVE:  2017-12-04 16:54:24.803306
Completed Indexing:  2017-12-04 16:54:59.303545
Finished At:  2017-12-04 16:55:00.160004
```

Figure 18 ***CYBOK Setup. The user enters their selections and the instance is built***

```
>>> cybok_wrapper.search_wrapper(cybok)
Set a threshold for filtering (floating point):0.5
Select query mode:-freetext
Query text: http query request in client server
1-GRAM RESULTS
CAPEC-22        Exploiting Trust in Client      3.1675541297255156
CWE-602         Client-Side Enforcement of Server-Side Security         2.750493504198467
CAPEC-207       Removing Important Client Functionality         2.61518193092608
CAPEC-202       Create Malicious Client         2.561600417798357
CWE-603         Use of Client-Side Authentication       2.2888302510010696
CWE-943         Improper Neutralization of Special Elements in Data Query Logic         2.2378069977219153
CAPEC-65        Sniff Application Code          2.1628834616011825
CAPEC-208       Removing/short-circuiting 'Purse' logic: removing/mutating 'cash' decrements    2.1280244347882595
CWE-444         Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')         2.045171009713962
CAPEC-220       Client-Server Protocol Manipulation     1.9712336420927483
CAPEC-14        Client-side Injection-induced Buffer Overflow   1.961960652757369
CAPEC-261       Fuzzing for garnering other adjacent user/sensitive data        1.9380252860030585
CWE-918         Server-Side Request Forgery (SSRF)      1.6751292330368646
CAPEC-33        HTTP Request Smuggling          1.6630727274204584
CAPEC-105       HTTP Request Splitting          1.6161640837489806
CAPEC-85        AJAX Fingerprinting     1.590535026561544
CAPEC-183       IMAP/SMTP Command Injection     1.5379884419366627
CAPEC-142       DNS Cache Poisoning     1.3823176393336785
CVE-2016-1914   CVE-2016-1914   1.3560133538191321
CVE-2015-1820   CVE-2015-1820   1.3424891421554344
CWE-437         Incomplete Model of Endpoint Features   1.3403492723349195
CAPEC-107       Cross Site Tracing      1.333612233607044
CAPEC-111       JSON Hijacking (aka JavaScript Hijacking)       1.3228164519836074
CAPEC-198       XSS Targeting Error Pages       1.304058276779367
CAPEC-73        User-Controlled Filename        1.3005973519915628
```

Figure 19 ***(a) At each iteration of the search wrapper, the user may enter a numeric threshold lower-bounding the relevance of any results presented. (b) A Free-text search is done by entering "-freetext", followed by the query string at the next prompt.***

The scores in the rightmost column of Figure 19 are the sum of the partial scores each entry accumulated from across all n-grams generated from the query. These are shown in more detail in Figure 20.

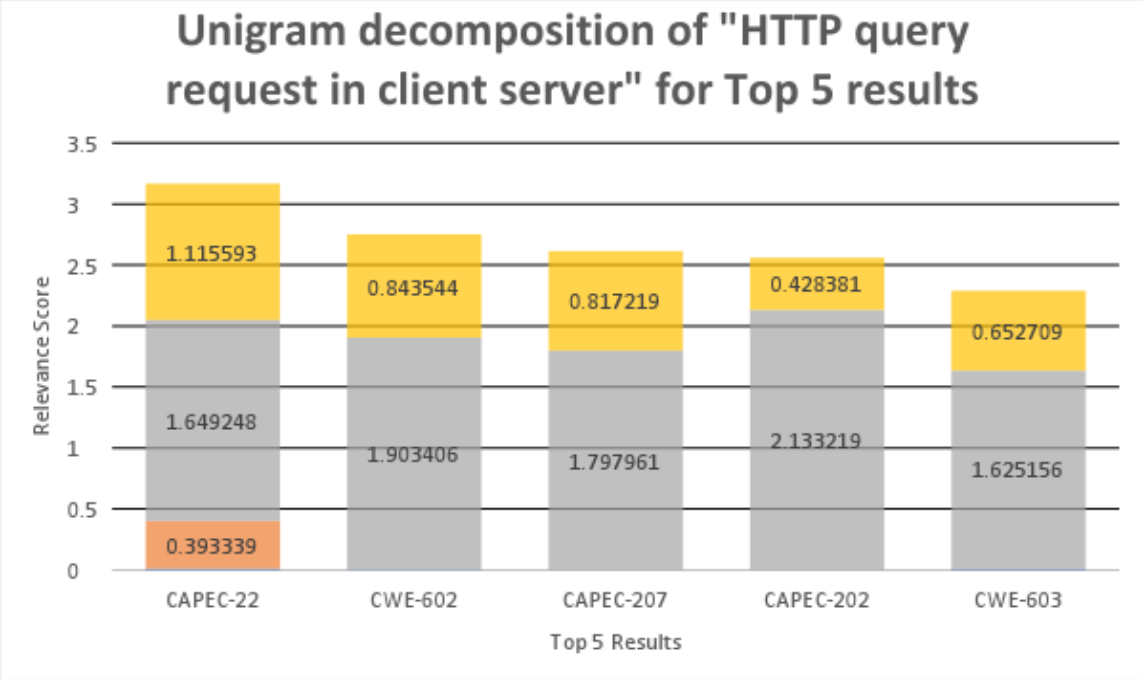


Figure 20 ***Shows the partial scores obtained for each unigram in the query from Figure 6.8. Any document which does not include these terms will not show up in any of the indices, and thus will not contribute to the latency of the CYBOK Search Engine.***

SUMMARY

Using CYBOK cyber repositories and search engine methods in a guided manner, they become a powerful resource in applying the MIssionAware posture to DoD systems. CYBOK is like an engine for Vulnerability Detection and Analysis Process, it brings pertinent and relevant attack information in the proper context to the system engineer and cyber analyst.

- CYBOK finds both known and unknown attack relations and patterns in various cyber vulnerability databases through advanced data analytics, Natural Language Processing (NLP), and graph search techniques.
- CYBOK allows system engineers to create relevant cyber-attack data with respect to the system, and rank the significance of the attack patterns to the system.

HYBRID THREAT MODELING METHOD

BACKGROUND – 2016 SEI RESEARCH PROJECT OUTCOMES

In our prior study we compared the use of 3 popular threat modeling methods: PnGs, STRIDE, and Security Cards, and found that threat models built using PnGs exhibited a higher degree of consistency than other techniques. Nonetheless, no individual threat model included all

identified threats. We therefore explored the idea of crowd-sourcing the task of threat identification. Our approach used information retrieval techniques to analyze the identified threats and collate them and provided the results to a human analyst to assist with construction of a unified threat model.

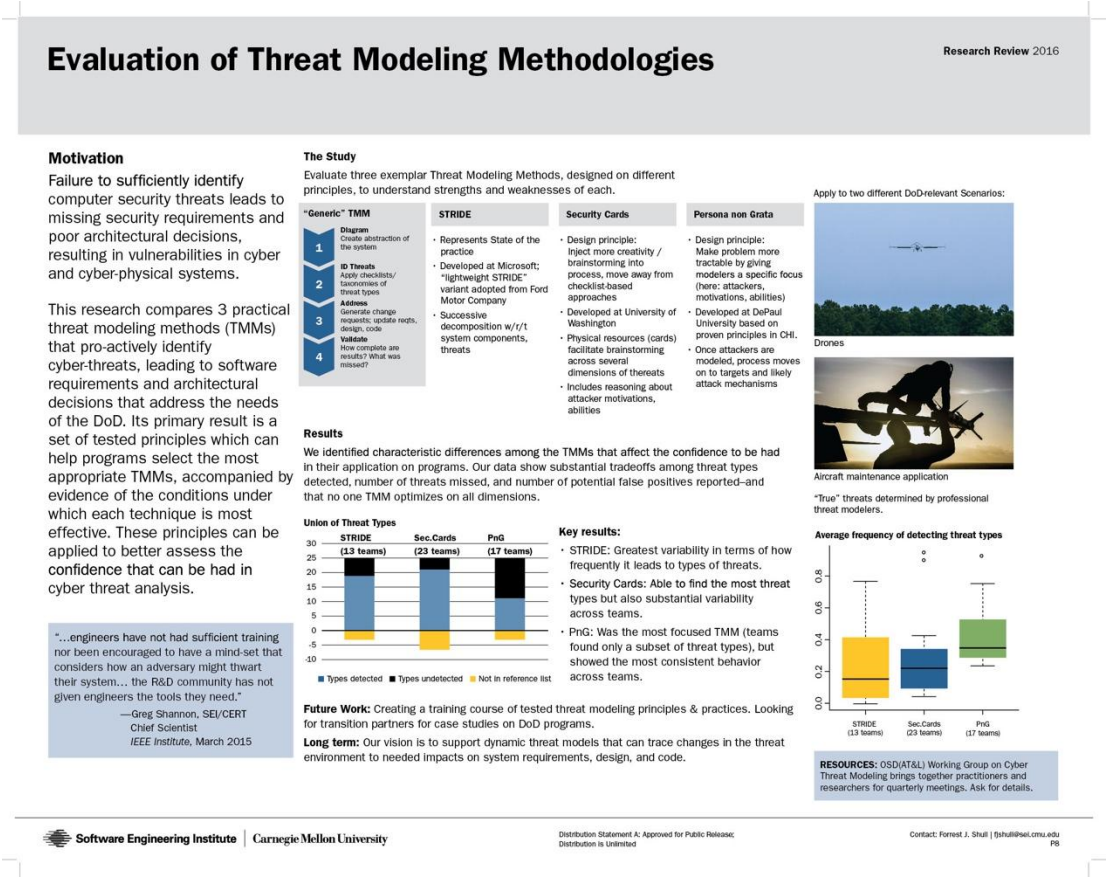


Figure 21 ***Evaluation of Thread Modeling Methodologies***

SEI 2017 DEVELOPMENT OF HYBRID THREAT MODELING METHOD

In developing the hTMM, we considered the following desirable characteristics for a Threat Modeling Method:

Desirable Characteristics for a Threat Modeling Method

- No false positives
- No overlooked threats
- Consistent results regardless of who is doing the threat modeling
- Cost-effective (doesn't waste time)
- Empirical evidence to support its efficacy

Other Considerations

- Has tool support
- Suggests a prioritization scheme
- Easy to learn, intuitive
- Encourages thinking outside the box
- Can be used by non-experts, or conversely, optimal for experts
- Clearly superior for specific types of systems

The steps in the hTMM are as follows:

1. Identify the system you will be threat modeling. Execute steps 1-3 of SQUARE or a similar security requirements method.
    a. Agree on definitions
    b. Identify a business goal for the system, assets and security goals
    c. Gather as many artifacts as feasible
2. Apply security cards in the following way, as suggested by the developers[8].
    a. *Distribute the Security Cards to participants either in advance or at the start of the activity.* Include representatives of at least the three following groups of stakeholders: System users/purchasers, system engineers/developers, and cybersecurity experts. You may find that within each of those categories, there are multiple distinct perspectives that need to be represented. Other relevant stakeholders can be included as well.
        i. System users/purchasers would include those purchasing or acquiring the system, end users, and other groups with a vested interest in the system. For example, in a scientific research organization, stakeholders could include the scientists conducting research, the executive directors of the organization, human resources, and information technologists managing the system. Each would have their own ideas about assets that need to be protected and potential attackers.
        ii. Cybersecurity experts could be part of a separate specialized team or integrated into the project team. They could include roles such as system administrators, penetration testers or ethical hackers, threat modelers, security analysts, and so on.
        iii. The engineer/development team members could range from systems engineers, requirements analysts, architects, developers, testers, and so on.
    d. Have the participants look over the cards along all four dimensions: Human Impact, Adversary's Motivations, Adversary's Resources, and Adversary's Methods. Read at least one card from each dimension, front and back.
    e. Use the cards to support a brainstorming session. Consider each dimension independently and sort the cards within that dimension in order of how relevant and risky it is for the system overall. Discuss as a team what orderings are identified. It's

[8] http://securitycards.cs.washington.edu/

important to be inclusive, so do not exclude ideas that seem unlikely or illogical at this point in time. As you conduct your brainstorming exercise, record the following:

i. If your system is compromised, what assets, both human and system, could be impacted?

ii. Who are the Personae non Gratae[9] who might reasonably attack your system and why? What are their names/job titles/roles? Describe them in some detail.
    1. What are their goals
    2. What resources and skills might the PnG have?

iii. In what ways could the system be attacked?
    1. For each attack vector, have you identified a PnG (or could you add a PnG) capable of utilizing that vector?

3. Once this data has been collected, you have enough information to prune those PnGs that are unlikely or for which no realistic attack vectors could be identified. Once this has been done, you are in a position to:
    a. Itemize their misuse cases. This expands on HOW the adversary attacks the system. The misuse cases provide the supporting detailed information on how the attack takes place.
4. Summarize the results from the above steps, utilizing tool support, as follows[10]:
    a. Actor (PnG): who or what instigates the attack?
    f. Purpose: what is the actor's goal or intent?
    g. Target: what asset is the target?
    h. Action: What action does the actor perform or attempt to perform? Here you should consider both the resources and the skills of the actor. You will also be describing HOW the actor might attack your system and its expansion into misuse cases.
    i. Result of the action: What happens as a result of the action? What assets are compromised? What goal has the actor achieved?
    j. Impact: What is the severity of the result (high, medium, or low)
    k. Threat type: (e.g., denial of service, spoofing)
5. Once this is done, you can continue with a formal risk assessment method, using these results, and the additional steps of a security requirements method such as SQUARE, perhaps tailoring the method to eliminate steps you have already accounted for in the threat modeling exercise.

[9] https://www.infoq.com/articles/personae-non-gratae

[10] https://resources.sei.cmu.edu/asset_files/Presentation/2016_017_001_474200.pdf

Steps 1 and 5 are activities that precede and follow the bulk of the threat modeling work. We felt it was necessary to include these, to understand where hTMM fits into lifecycle activities, specifically security requirements engineering.

Our initial thought was that we could apply security cards and then PnG to come up with a hybrid method. It became clear to us that we could not just do security cards followed by PnG, as each one is a threat modeling method in its own right. Therefore, we needed to consider specific aspects of both security cards and PnG, and also consider what we had learned by our experiments with STRIDE. We wanted to capture the best features of all the models. Having said that, it's possible that the hTMM could change after we have done a few medium to large pilot threat modeling projects, in addition to the small case studies in this report, not to mention those conducted by students on our earlier research project.

Ideally, when considering stakeholders, we would like to be able to recommend a core set of important perspectives (a team might add others that are important in context, but should not do this without all of the core being represented). Possibly this could be done by suggesting a particular focus that we think each perspective brings. It's worth noting that some pilot studies will probably determine whether this is feasible, or whether the needed perspectives vary too widely depending on the system being studied to recommend a core set of perspectives.

We visualized the hybrid approach as using Security Cards to cast the net wide and generate lots of ideas, and then PnG to filter and weed out the poor ones. We do have a concern about possible overlap between steps 2 and 3. After a few pilots, we may conclude that we are doing the same thing twice, or we may decide that the high-level and refined views of the threats are both necessary. Looking at it another way, using PnG in conjunction with attack vectors used by a persona non grata could serve as a confirmation and sanity check for the work done with security cards, and vice versa.

For step 4, we felt that tool support would help to organize the results and eliminate the need for what otherwise appeared to be a copy and paste exercise. Ultimately, it's possible that more sophisticated tools could help with the analysis, but at a minimum tools should be used to eliminate mundane transcription and bookkeeping tasks.

MEASUREMENT CONSIDERATIONS

1. From a research point of view, we would like to collect data on number and types of issues that come from each stakeholder type so we could have some evidence about what each contributes to the overall threat model. However, we are not sure if we can collect at that level of granularity.

2. Building on the comments above, it would be good to know how many items get generated in Step 2, and then how many are dropped vs. refined in Step 3? With some work we could also map those to an oracle dataset so that we could see if the ones that got filtered were actually related to real threats or if the ones that get refined in PnG were false positives that weren't worth the effort.

We applied the hTMM to a drone scenario. We itemize the results of applying the method in the following sections, along with recommendations for improvement. The numbering below refers to the steps in the hTMM method described previously.

1 System Info Gathering

The system to be studied is a drone or drone swarm that is on its way to deliver emergency supplies to flood-affected populations. It is dispatched by a team consisting of local government authorities and its drone technology contractors. See Figure 22 for an example of a drone swarm.

The drones face several potential threats; both physical and cyber in nature. We will consider potentially likely scenarios of drone attack and how those attacks affect the people who depend on the drones and drones themselves.

Figure 22 An example image of a drone *swarm*[11]

2. Brainstorming

2a) Involve Representative Stakeholders

The following is a ranking of various stakeholders based on the value they would add to a brainstorming session.

[11] Image source - http://www.ioti.com/security/drones-are-coming-take-cover

Stakeholder	The Expertise	Value Added
Drone Designer	Knows the safe operating range and the breakdown range its dynamics, the design specifications of each component and the drone.	High, because the information can be used to analyze both cyber threats that try to drive the drone out of its operational range and physical threats where the components fail to propel the drone in the desired path.
Drone Pilot	Knows the optimal parameter settings for safe navigation, and can program the drone to go from point A to point B without violating the drone flying regulations, if any.	High, because the knowledge of the potential routes the drone may take can be used by attackers to capture or damage the drone. This should help the analysts to learn about the locations where the attackers can intercept the drone.
Telecommunications Expert	Given a drone make and communication electronics used, this person can give information on the type signals and their adversaries during a signal based attack such as electromagnetic jamming.	High, because jamming of electronics signals is a known and viable attack pathway.
Local government official	Knows about the drone use cases for relief and other public service missions.	Low, because, we can learn about the load carried by drones and dates of use cases, but not any technical information about the drone itself.

Threat modeler	Knowledgeable to identify new attack use cases or evaluate the attack model.	High, because this person can refine threat modeling and may contribute with alternatives.

2b) Review the Following Threat Model Dimensions

The dimensions reviewed include:

- the dimension of human impact
- the dimension of threat to drone body and its electronics (GPS and Telecom)
- the dimension of threat motivation
- the dimension of attack channels
- the dimension of adversary's resources
- the dimension of adversary's methods
- the dimension of design risk as a whole
- the dimension of impact
- the dimension of financial loss and the cost of potential investigations

Part 1: Preform ranking within each category and why it is considered a potential threat:

Human Impact Cards:

1. Emotional wellbeing (those suffering from the disaster are deprived of basic commodities and get depressed-the primary subject of the threat)
2. Physical Wellbeing (health is affected due to lack of timely food supplies and medicine-the primary subject of the threat)
3. Relationships (the relations between the people, local authorities and government is at stake if the rescue mission fails-a secondary subject)
4. Unusual impacts (loss of property, loss of life, loss of trust in local government, loss of businesses-a secondary or ternary subject)

Adversary's Motivations:

1. Money (the goods stolen from the drones and the drones/components can be resold to make money- the profitable nature makes this rank 1!)

2. Warfare (some local trouble makers may see this as a route and non-violent means of attack-ease of attack i.e., without having to face other humans in the operations makes this an above the rest)

3. Politics (oppositions and opposition groups may intrude to bring bad name to local government – can lead to change of power, so it becomes attractive)

4. Unusual motivations (hack the drones and use them for other unauthorized purposes such as flying in restricted zones or delivery of harmful goods or simply destroy)

Adversary's Resources:

1. Expertise (the attacker has all the expertise to hack the brand of drones that are used in the mission)

2. Inside Knowledge (access to inside knowledge makes the attack viable)

3. Money (money flowing in for political reasons to bring down the reputation)

4. Inside Capabilities (an insider who turns an attacker can do a lot of damage to the drones)

Adversary's Methods:

1. Physical Attack (shoot the drone with a drone gun)

2. Technological Attack (jam the GPS or the rotors)

3. Multiphase Attack (damage partially, like one rotor and partially disable the drone to take into control)

4. Manipulation or Coercion (hack the drone information system and its GPS, then change the destination or send it back to the origin or make it loose the sense of direction)

Part 2: In-depth analysis of all the potential threats:

Human Impact Cards:

1. Emotional wellbeing (those suffering from the disaster are deprived of basic commodities and get depressed-the primary subject of the threat)

2. Physical Wellbeing (health is affected due to lack of timely food supplies and medicine-the primary subject of the threat)

3. Relationships (the relations between the people, local authorities and government is at stake if the rescue mission fails-a secondary subject)

4. Unusual impacts (loss of life, loss of trust in local government, loss of businesses-a secondary or ternary subject)

Type	Actor	Action	Target	Purpose	Result	Impact
denial	attacker	Attack methods 1-4	Drone (physical, cyber)	Motivations 1-4	Human Impacts 1-4	1-High 2-High 3-Low 4-Low

Adversary's Motivations:

1. Money (the goods stolen from the drones and the drones/components can be resold to make money- the profitable nature makes this rank 1!)
2. Warfare (some local trouble makers may see this as a route and non-violent means of attack-ease of attack i.e., without having to face other humans in the operations makes this an above the rest)
3. Politics (oppositions and opposition groups may intrude to bring bad name to local government – can lead to change of power, so it becomes attractive)
4. Unusual motivations (hack the drones and use them for other unauthorized purposes such as flying in restricted zones or delivery of harmful goods or simply destroy)

Type	Actor	Action	Target	Purpose	Result	Impact
1-Capture 2,3,4-hack	attacker	Intrusion	drone	misuse drone	Adversary's Motivations 1-4	1-High 2,3,4-Low

Adversary's Resources:

1. Expertise (the attacker has all the expertise to hack the brand of drones that are used in the mission)
2. Inside Knowledge (access to inside knowledge makes the attack viable)
3. Money (money flowing in for political reasons to bring down the reputation)

4. Inside Capabilities (an insider who turns an attacker can do a lot of damage to the drones)

Type	Actor	Action	Target	Purpose	Result	Impact
Denial, Spoofing, Jamming, Screening	Tech Expert Hacker	Cyber attacks	Drone-physical, GPS, accelerometer, camera	Adversary's Motivations 1-4	Human Impacts 1-4	1,2,3,4 High
Denial, Spoofing, Jamming, Screening	Specific Attacker: A Nation	Physical, cyber attacks	Drone-physical, GPS, accelerometer, camera	Adversary's Motivations 1-4	Human Impacts 1-4	1,2,3,4 High
Screening, Shooting	Specific Attacker: A Gangster	Physical attacks	Drone, Any of its components	Adversary's Motivations 1-3	Human Impacts 1-4	1,2,3,4 High

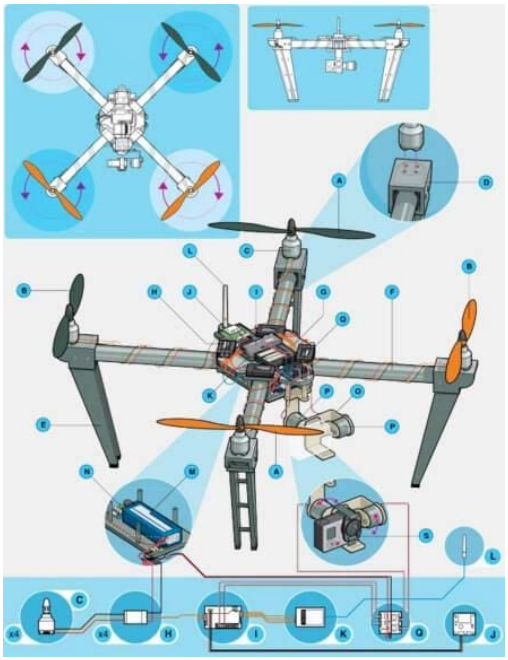


Figure 23 Example of Drone *Components*[12].

Adversary's Methods:

To analyze how a drone can be subjected to an attack, let us consider an example drone. Figure 23 shows typical drone parts: Propellers, Brushless Motors, Motor Mount, Landing Gear, Boom, Drone Body Part, Electronic Speed Controllers, Flight Controller, GPS Module, Receiver, Antenna, Battery, Battery Monitor, Gimbal, Gimbal Motor, Gimbal Control Unit, Camera, Sensors and Collision Avoidance Sensors. The attack can in principle be on any of the components. We consider a few most likely cases.

1. Physical Attack (shoot the drone with a drone gun, direct objects or spray a dark paint on drone's camera to blind the drone)
2. Technological Attack (jam the GPS or the propellers)
3. Multiphase Attack (damage partially, like one propeller and partially disable the drone to take into control)
4. Manipulation or Coercion (hack the drone information system and its GPS, then change the destination or send it back to the origin or make it loose the sense of direction)

[12] Image Source: https://www.dronezon.com.

Type	Actor	Action	Target	Purpose	Result	Impact
damaging, capturing, redirecting	attacker	shooting, hacking, modifying, parametes	drone, physical, GPS, accelerometer, computer	Adversary's Motivations 1-4	Human Impacts 1-4	1,2,3,4 High

2c) Brainstorm with Attention to Mal Actors

Mal Actors (Personae non Gratae):

The insiders:

-drone physical experts, drone software experts, drone operational support staff

The outsiders:

-individuals who may think that the drone is harmful to them if it is flying near by

-expert drone hackers with some of the motivations listed above

3. Prune Unlikely/Incomplete PnGs

Summarize the critical PnGs

PnG	Type of Treat	Threat Risk
Drone Hacker	Cyber attack or jamming telecommunication	High, the drone may malfunction and fail the mission
Drone Pilot	Reroute the drone	High, due to loss of drone or loss of control of navigation

Drone Pirate	Capture the drone when it is closer to ground for deliveries	High, because the drone will have to make a landing to deliver goods in remote areas during relief operations.
Drone Reverse Engineer	If the drone goes through a supply chain attacker during manufacturing or when it is captured, this person can modify the components or their functions	Low, this requires supply chain attack and reengineering expertise, which is not a common skill during capture. But this can be High if it happens during manufacturing of the drone.

4) Key Threats

(1) Cyber attack on drone autonomous navigation programming

(2) Jamming of drone electronics such as GPS and telecommunications.

4a) Actor: Drone Hacker

4b) Purpose: To disable the drone or its mission

4c) Target: The drone

4d) Attack Method: Technological Attack, Multiphase Attack

4e) Results: Damaged relationships; public-government, drone manufacturer-local officials

5) Final Assessment

5a) Type of Impact	5b) Financial Loss (Rough Estimates)	5c) Social Disorder
Failed mission of the drone	If the mission fails, then an alternate attempt for deliveries will be required. This may require exploring alternate	Human Impacts: Emotional wellbeing Physical wellbeing Relationships

	resources such as helicopters and result in more unplanned expenditures. The loss depends on the number of days of the mission. Upper limit ~$1 Million	Unusual impacts
Loss of drone navigation control	If a few drones are lost, then replacement drones needs to be dispatched or different make of drones needs to be dispatched to fulfill the mission. This may cost a few thousands of dollars. Upper limit ~ $20,000	Human Impacts (as listed above). If drones fly or land in areas other than intended locations, the residents there may be distracted or affected leading to social disturbance.
Loss of drone or its supplies	The cost of the entire fleet of drones. It depends on the make and the model of drone. Upper limit ~ $200,000	Human Impacts
Drone actions do not correspond to initially programmed commands	Need to hire more manpower to locate the drone, retrieve and hours of programming to realign the commands. This leads to additional spending. Upper limit ~$50,000	Human Impacts. Disorder in the drone technology department, work overload on employees and potential delays in future missions.
Replaced or reengineered drone components	If the drone is experiencing a supply chain attack, then it requires physical component diagnostics and replacing with true components. Upper limit ~$100,000	Human Impacts. Disorder in the drone technology department at the management level due to the poor choice of drone manufacturers.

FINAL THOUGHTS FOR FURTHER MODEL AND TOOL DEVELOPMENT

The following are the lessons learned from the perspective of modelers. These can be considered as a passive component of the model and may help a user to expand the scope of modeling as needed during applications.

1. Specific information on the type of drone, a geographic location of the mission, the budget available for the missions would be further helpful parameters to fine tune the model for specific use cases.
2. For a more efficient and routine application of the model, a Windows-Based Tool that can be used to build the model, for example using toolbar options such as "Select from a list of Icons" or "Dropdown lists" to add a piece of the model would be very helpful.
3. Application of the model to other example devices should be highly encouraged for cross-culturing of ideas. Such efforts can lead to hTMM model refinement and advancement to a higher level of complexity.

PUBLICATIONS

Bakirtzis, G., Carter, B. T., Elks, C. E., & Fleming, C. H. (2018). A Model-Based Approach to Security Analysis for Cyber-Physical Systems. *Under Review.*

Carter, B. T., Bakirtzis, G., Elks, C. R., & Fleming, C. H. (2018). A Systems Approach for Eliciting Mission-Centric Security Requirements. *Under Review.*

Bakirtzis, G., Carter, B. T., Elks, C. E., & Fleming, C. H. (2018). MISSION AWARE: Evidence-Based, Mission-Centric Cybersecurity Analysis. *Under Review.*

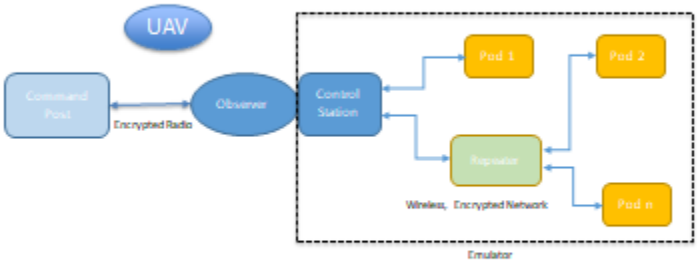
Purpose

Use an example of an emulated Army system to highlight system design decisions and technologies related to Cyber Physical Security.

3

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

System Description



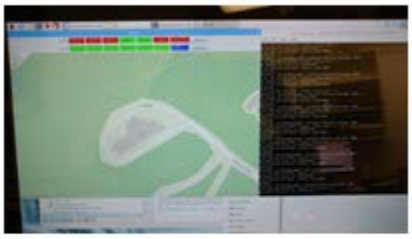
4

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

Emulator Description

- Control Station (CS)
  - Graphical User Interface (GUI)
    - Map showing location of CS and pods
    - Status of each pod (enabled, connected, LEDs)
    - Actions (enable/disable, buzzer, LEDs)
  - Communications server
  - Wireless
  - Encryption
- Pod
  - Communications client
  - Wireless
  - Encryption
  - LEDs and sensors
- Repeater
  - Relay messages between CS and pods
  - Wireless

Emulator Demonstration



Constraints

- Limited information on the Army system
- Low or no cost
  - Software: develop or open source
  - Hardware: low cost or on-hand
- Limited knowledge of developers

Implementation: Hardware

- Raspberry Pi 2 kit (~ $80)
- Raspberry Pi 2 Single Board Computer
  - 900 MHz quad-core ARM Cortex A7
  - 1 GiB RAM
  - HDMI, USB, Ethernet
  - 40 General Purpose I/O
- WiFi adapter

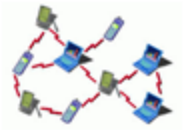
Implementation: Operating System

Raspbian

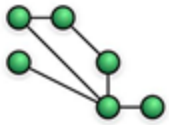
- Raspbian (Linux Debian)
  - Pixel desktop for CS
  - Lite for pods and repeater (no desktop)
- Advantages:
  - Prior experience
  - Community support
  - Numerous open source packages

Implementation: Wi-Fi Architecture

Infrastructure



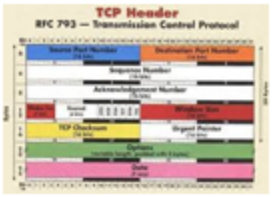
Ad-hoc



Mesh

- Mesh
  - IEEE 802.11s
  - Included in Raspbian
  - No additional software for repeater
  - Easy setup
  - Automatically establishes links to peers
  - Automatic routing

Implementation: Network Protocol



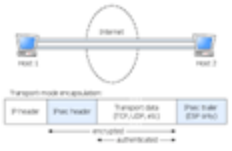
- User Data Gram Protocol (UDP)
  - Unreliable connection (link-to-link)
  - Data can be lost or duplicated
  - Data can arrive out of order
  - Faster than IP
- Internet Protocol (IP)
  - Reliable connection
  - No data loss or duplication
  - Data received in same order sent
  - Slower than UDP

Implementation: Data Protocol

State: HSP#L######N######E#Z#C#G<latitude>,<longitude>,<altitude>\r\n

Heartbeat: HBP#r\n

Implementation: Encryption



- Internet Protocol Security (IPsec)
  - Authenticates and encrypts data packets
  - Negotiation of cryptographic keys
  - Mutual authentication of hosts
  - Host-to-host mode
- strongSwan:
  - Easy to install, learn, and configure
  - Multiple encryption algorithms
  - Open source

Implementation: CS GUI

- Interface to operator
- Part of CS application (GUI and server)
- Coded in Java
- Control interface
  - Status of each pod
  - Actions: enable/disable, LED on
- Map
  - Shows positions of pods
  - OpenStreetMap
  - Needs internet connection or cached map tiles

Implementation: CS Communications Server

- Central communications for pods and CS GUI
- Part of CS application (GUI and server)
- Coded in Java (sockets)
- Each pod connects to server (thread per connection)
- GUI receives data from server via message queues

Implementation: Pod Communications and I/O

- Pod communications
  - Receives commands from CS, sends confirmation
  - Sends heartbeat
- I/O
  - Changes color of enable/disable LED
  - Connection LED on/off
  - Action LEDs on
- Coded in Python

Testing Tools

- WireShark
  - Examine Ethernet traffic
  - Examine message content (unencrypted)
  - Verify message encryption
- IPsec
  - Verify connections
  - Verify encryption mode
- Mesh utilities
  - lw (station status and paths)
  - ping (test communications)
  - arp (show connections)

Software Requirements Specification

Georgios Bakirtzis
bakirtzisg@ieee.org
Tue 7 Nov 2017
Updated: Thu 9 Nov 2017

1 Overview

This section gives the scope description and overview of everything included in this Software Requirements Specification (SRS) document.

1.1 Purpose

This SRS captures the general software requirements for the Cyber Analyst Dashboard. Through these requirements we produce a functional specification for the Cyber Analyst Dashboard including basic functionality, features, and interface.

1.2 Scope

The Cyber Analyst Dashboard is a unified view between the system model and its associated attack vector space. Through this view a cyber analyst should be able to navigate the potentially mission-critical parts of the system architecture, how the associate to the specified expected behavior, and finally how they relate to the high-level mission requirements.

2 Overall Description

This section gives an overview of the system:

1. what service it is expected to provide to cyber analysts;
2. how it is expected to be used;
3. what it requires as prerequisites to operate correctly; and
4. how it is (as of now) going to be implemented.

2.1 Product Perspective

By using the Cyber Analyst Dashboard the cyber analyst should have a comparable view to the systems attack vector space, including potential attack patterns, system weaknesses, and recorded vulnerabilities for the specific software run on the system. By merging these views the cyber analyst should be able to inform himself about the security posture of the system holistically and either find architectural preemption or mitigation techniques to address the

attack vectors he deems important. This way he can better inform and communicate with system designers that then can provide architectural preemption and mitigation strategies or craft concrete system requirements to address these issues. Indeed, through this tool the cyber analyst should be able to communicate his findings to other stakeholders, for example system designers and operators, by presenting a common view through the model and its associated attack vector space.

2.2 Product Functions

The major functions of the Cyber Analyst Dashboard are:

1. Import functions for GraphML files.
2. Visualization functions for the associated graphs.
3. Interactivity functions for manipulating the graph and presenting different visualization information to the cyber analyst.
4. Search functions to refine the graphs to deal with their intrinsic complexity.
5. Addition function to add cyber analyst specific information, e.g., adding an attack vector to the attack vector space graph that might not be included in the databases but the analyst knows it to exist within the system structure and associate it to a specific architectural element.
6. Trace functions for the mission specification graph.

2.3 User Characteristics

The Cyber Analyst Dashboard is expected to be used by academic researchers to find novel approaches to visualization mission specification and its associated attack vector space. Additionally, it is expected to be used by actual cyber analysts to evaluate its efficacy in finding possible vulnerabilities in the architecture of the system and how they relate to higher-level mission requirements.

2.4 Design and Implementation Constraints

The product implementing the Cyber Analyst Dashboard will be standalone given the correct information as described in the following section. The software is expected to be improved (either by removing or adding features based on the “field tests”) throughout its lifecycle and be robust enough to illustrate its utility to stakeholders. The software is assumed to be used by people that have some background in security of cyber-physical systems. This means we are not targeting novices in the field but seasoned researchers. This allows us to be more specific with the definitions of what is an architecture and what is an attack that can apply to that architecture and how that architecture relates to its behavioral representation and accordingly to its mission requirements.

2.5 Assumptions and Dependencies

The main assumption made with regard to the use of the Cyber Analyst Dashboard is that the user has produced two GraphML files: (i) a mission specification graph and (ii) an attack vector space graph produced for the architectural part of the mission specification.

Currently, the mission specification graph in MissionAware is constructed in a SysML requirements diagram. First we model top-to-bottom by producing mission requirements in SysML requirements diagram, a behavioral description in SysML activity diagrams, and the architectural specification in SysML block definition diagrams and internal block diagrams. On the second pass we produce the mission specification by extending the initial mission requirements diagram to include further requirements identified through the top-to-bottom modeling, how the associate to the corresponding behavior, and finally which architectural elements are mission-critical based on the behavior. By constructing this trace between the three domains we create the mission specification graph. While a SysML package would be needed to faithfully follow the methodology the Cyber Analyst Dashboard doesn't require a SysML implementation of the model but rather just a GraphML representation of the mission specification (wherever it comes from).

The attack vector space graph is assumed to be created by finding vulnerabilities associated with the architectural model through open attack vector databases, for example, CAPEC, CWE, CVE. This attack vector graph should already contain which vulnerabilities are associated with which architectural element, e.g., spoofing attack as described in CAPEC entry 148 is associated with the NMEA GPS provided in the architectural model. In the MissionAware methodology this shall be provided by the Cyber Body of Knowledge (CYBOK) tool, but again this is not a requirement for the Cyber Analyst Dashboard to operate correctly, merely a GraphML file that contains that information.

2.6 Developmental Tools

There are three main library options for implementing the software defined in this SRS:

1. D3.js (JavaScript) & cola.js (JavaScript)
2. cytoscape.js (JavaScript)
3. plotly (Python)

From the three the most seasoned is the first, D3.js, which offers a force-directed graph layout. An extension of D3.js, cola.js, allows for constrained force-directed layout, which could allow an automated visualization of the three domains in the mission specification. The second, cytoscape.js, provides a wide selection of graph drawing algorithms. This follows naturally since cytoscape (both as standalone software and JavaScript API) is geared only towards graph structures. Having said that, cytoscape.js is a new, rapidly changing API (albeit somewhat more stable than a year ago when this question first arose). Finally, the newest and least seasoned API is provided by plotly. This library is also restricted to only certain graph drawing algorithms, since its purpose is to be more general and more idiomatic than either of the above options.

The use of more than one API is possible but not recommended since the libraries can have the same dependencies but not necessarily relying on the same version (this is mostly true for the first two libraries that are based on JavaScript, integrating a python library with a JavaScript library would be a task in and of itself).

Another possible developmental tool to consider is react.js. A library that allows a user to programmatically modify DOM elements (which all three above libraries can above can do) and provides smoother interactivity than either of the visualization libraries presented in this SRS.

(*The current recommendation is to utilize cytoscape.js for graph visualization and react.js to access and manipulate DOM elements. This is, however, still in the evaluation phase.*)

In the future, electron allows us to implement a cross-platform standalone application based on the code that was run in the browser during development.

2.7 User Documentation

Programming language (that is JavaScript) documentation:

- JavaScript reference https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Developmental tool documentation:

- d3.js documentation https://github.com/d3/d3/wiki
- cytoscape.js http://js.cytoscape.org/
- plotly API reference https://plot.ly/api/

Graph format (that is GraphML) documentation:

- The GraphML file format http://graphml.graphdrawing.org/
- GraphML format (from Gephi) https://gephi.org/users/supported-graph-formats/graphml-format/

3 Software Specific Requirements

This section contains all of the specific functional requirements and features of the Cyber Analyst Dashboard. It gives a detailed description of the system and all of its features individually.

3.1 External Interface Requirements

This section provides a detailed description of all inputs to and outputs from the system and provides the basic information necessary to the understanding of the user interface.

3.1.1 User Interface

The scope of this SRS limits the user interface of the Cyber Analyst Dashboard to include only the interface between the cyber analyst and the information produced by the CYBOK tool and mission specification model respectively. It does not consider for example the interface between the analyst and the model or how the analyst will produce a set of applicable attacks given a system architecture model. Hence, the user interface included only the visualization and interactivity functions provided by the Cyber Analyst Dashboard to better reason around the system's structure and its corresponding attack vector space.

3.1.2 Software Interfaces

The Cyber Analyst Dashboard is operating system agnostic since it is run in the browser. As has been indicated before Electron could be used to construct a cross-platform standalone application when the implementation is more mature. The software assumes a current browser version since it used elements of recently constructed and implemented web browser standards, including at a minimum svg support and ES6 style JavaScript.

3.2 Functional Requirements and Implementation Details

This section includes the requirements that specify all the fundamental actions of the software system. The functional requirements are divided into (i) general functional requirements and (ii) implementation details.

3.2.1 General Functional Requirements

3.2.1.1 Functional Requirement 1

ID: FR1
TITLE: Parsing Graph Files
DESC: The software must be able to import and parse GraphML files to be able to visualize the information contained within them.

3.2.1.2 Functional Requirement 2

ID: FR2
TITLE: Graph Visualization
DESC: The software must be able to present the graph files using a graph visualization algorithm. The graph visualization can take a lot of forms but a requirement for the mission specification is that it needs to constraint the nodes of each domain in their individual domains, for example, mission requirements should be constrained at a higher level than behavior and architecture, thus achieving a hierarchy for the mission specification that is already present in the model.

3.2.1.3 Functional Requirement 3

ID: FR3
TITLE: Graph Interactivity

DESC: The software must allow the cyber analyst to interact with the graph. This would include functions like tooltips to present more information at the vulnerability space or tooltips to see the attributes of a given architectural element or showing the full text of a mission requirement. The cyber analyst might also want to add his own set of vulnerabilities to the attack vector space and associate them to an architectural element in the mission specification. Additionally, the cyber analyst might want to search through the full space of vulnerabilities that might not have been automatically associated by CYBOK but he knows are applicable to the architectural description and should be taken into consideration for the assessment of the system's security posture.

3.2.1.4 Functional Requirement 4

ID: FR4
TITLE: Dual Graph View
DESC: The software should give a comparison view, for example, by presenting the attack vector space on the left and the mission specification on the right.

3.2.1.5 Functional Requirement 5

ID: FR5
TITLE: Traceability
DESC: The software must be able to present the full trace of any given element in the mission specification. For example the cyber analyst might want to assess the impact of a violated architectural element to the behavioral and mission domains respectively. In this case the software should color the trace on all domains. This is not strictly applicable to the architecture, any element if clicked should present its full trace to the other two domains through interactive functions (FR3).

3.2.1.6 Functional Requirement 6

ID: FR6
TITLE: "Linting" Rules
DESC: When the software is mature and a number of systems have been analyzed the software should be able to present the cyber analyst with preemption and mitigation techniques to a new system based on the results of older analyses. Linting in this case takes the software engineering term, where a fragment of code is assessed for best practices and provides warning to the programmer.

3.2.1.7 Functional Requirement 7

ID: FR7
TITLE: Filtering Capabilities
DESC: The software should be able to filter any graph for more specific information. For example, filter the attack vector space only to the subgraph associated with an individual architectural element or a collection of architectural elements.

3.2.2 Implementation Details

3.2.2.1 Implementation Detail 1

ID: ID1
TITLE: Data Conversion
DESC: The use of the command-line interface in cytoscape might be needed to convert a GraphML file to a D3.js compatible JSON file, which is not standard JSON. **This is only the case if D3.js is used, cytoscape.js has import capabilities directly from a GraphML file.**
DEP: Cytoscape

APPENDIX C: CITED AND RELATED REFERENCES

Allodi, L., Massacci, F., & Williams, J. The work-averse cyber attacker model: Theory and evidence from two million attack signatures.

Bakirtzis, G., Carter, B. T., Elks, C. E., & Fleming, C. H. (2018). A Model-Based Approach to Security Analysis for Cyber-Physical Systems. *Under Review.*

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993-1022.

Brandes, U., Eiglsperger, M., Lerner, J., & Pich, C. (2013). *Graph markup language (GraphML)* (pp. 517-541).

Brunner, M., Huber, M., Sauerwein, C., & Breu, R. (2017, July). Towards an integrated model for safety and security requirements of cyber-physical systems. In *Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on* (pp. 334-340). IEEE.

Burmester, M., Magkos, E., & Chrissikopoulos, V. (2012). Modeling security in cyber–physical systems. *International journal of critical infrastructure protection*, *5*(3), 118-126.

Chen, B., Kalbarczyk, Z., Nicol, D. M., Sanders, W. H., Tan, R., Temple, W. G., Tippenhauer, N. O., Vu, A. H., & Yau, D. K. (2013, December). Go with the flow: Toward workflow-oriented security assessment. In *Proceedings of the 2013 workshop on New security paradigms workshop* (pp. 65-76). ACM.

Corbett, J., & Crookall, J. R. (1986). Design for economic manufacture. *CIRP Annals-Manufacturing Technology*, *35*(1), 93-97.

Davis, K. R., Davis, C. M., Zonouz, S. A., Bobba, R. B., Berthier, R., Garcia, L., & Sauer, P. W. (2015). A cyber-physical modeling and assessment framework for power grid infrastructures. *IEEE Transactions on Smart Grid*, *6*(5), 2464-2475.

Frola, F. R., & Miller, C. O. (1984). System safety in aircraft management. *Logistics Management Institute, Washington DC*.

Griffiths, T. L., Jordan, M. I., Tenenbaum, J. B., & Blei, D. M. (2004). Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems* (pp. 17-24).

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media.

Kordy, B., Piètre-Cambacédès, L., & Schweitzer, P. (2014). DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer science review*, *13*, 1-38.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79-86.

Kutz, M. (Ed.). (2015). *Mechanical Engineers' Handbook, Volume 1: Materials and Engineering Mechanics*. John Wiley & Sons.

Hause, M. (2006, September). The sysml modelling language. In *Fifteenth European Systems Engineering Conference* (Vol. 9).

Hu, F. (2013). *Cyber-physical systems: integrated computing and engineering design*. CRC Press.

Lafferty, J. D., & Blei, D. M. (2006). Correlated topic models. In *Advances in neural information processing systems* (pp. 147-154).

Leveson, N. (2011). *Engineering a safer world: Systems thinking applied to safety*. MIT press.

Mcauliffe, J. D., & Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems* (pp. 121-128).

Mead, N. R., & Woody, C. (2016). *Cyber Security Engineering: A Practical Approach for Systems and Software Assurance*. Addison-Wesley Professional.

Mead, N. R., Morales, J. A., & Alice, G. P. (2015). A method and case study for using malware analysis to improve security requirements. *International Journal of Secure Software Engineering (IJSSE)*, *6*(1), 1-23.

National Research Council. (2015). *Interim report on the 21st century cyberphysical systems education*. Tech. Rep.

Nicol, D. M., Sanders, W. H., & Trivedi, K. S. (2004). Model-based evaluation: from dependability to security. *IEEE Transactions on dependable and secure computing*, *1*(1), 48-65.

Saravi, M., Newnes, L., Mileham, A. R., & Goh, Y. M. (2008). Estimating cost at the conceptual design stage to optimize design in terms of performance and cost. *Collaborative product and service life cycle management for a sustainable world*, 123-130.

Sun, C., Ma, J., & Yao, Q. (2016). On the architecture and development life cycle of secure cyber-physical systems. *Journal of Communications and Information Networks*, *1*(4), 1-21.

Ward, G. L., Bakirtzis, G., & Klenke, R. H. (2014). A Modular Software Platform for Unmanned Aerial Vehicle Autopilot Systems. In *52nd Aerospace Sciences Meeting* (p. 1050).

Ward, G. L. (2014). *Design of a small form-factor flight control system*. Master's Thesis. Virginia Commonwealth University.

Weaver, G. A., Cheh, C., Rogers, E. J., Sanders, W. H., & Gammel, D. (2013, November). Toward a cyber-physical topology language: Applications to NERC CIP audit. In *Proceedings of the first ACM workshop on Smart energy grid security* (pp. 93-104). ACM.

Young, W., & Leveson, N. (2013, December). Systems thinking for safety and security. In *Proceedings of the 29th Annual Computer Security Applications Conference* (pp. 1-8). ACM.