SYSTEMS ENGINEERING RESEARCH CENTER

Model-based Engineering for Functional Risk Assessment and Design of Cyber Resilient Systems

Technical Report SERC-2019-TR-002

February 22, 2019

Principal Investigator:

Dr. Peter Beling, University of Virginia

Co-Principal Investigator:

Dr. Barry Horowitz, University of Virginia

Dr. Cody Fleming, University of Virginia

Research Team:

Stephen Adams, University of Virginia

Georgios Bakirtzis, University of Virginia

Bryan Carter, University of Virginia

Tim Sherburne, University of Virginia

Carl Elks, Virginia Commonwealth University

Aidan Collins, Virginia Commonwealth University

Brandon Simon, Virginia Commonwealth University

Sponsor: DASD(SE)

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, TO#0094.

Table of Contents

List of Figures

List of Tables

Executive Summary

This report describes a 12-month research activity with the principal objective of continuing development, testing and evaluation of a methodology and supporting suite of model-based engineering tools for functional risk assessment and design of cyber resilient systems. Research tasks were structured to extend the methods and support tools for the decision problem of selecting defense and resilience methods in the design and modification of cyber-physical systems. Research reported here continues the efforts of previous SERC projects, notably RT-156 and RT-172, and leverages and contributes to contemporaneous work in RT-191. The project was carried out as part of an ongoing research partnership between the University of Virginia (UVA) and Virginia Commonwealth University (VCU). The UVA team led development of methods and tools to model the consequences of cyber attacks on cyber-physical systems, and the VCU team led development of tools that relate consequences to likely attacks.

Outcomes this year include developing a deeper understanding of open source databases of historical cyber attacks (e.g., CAPEC, CWE, CERT, and CVE), as well as defining and developing 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) development of the STRAT toolset to support CSRM and dynamic assessment of attack consequence, (2) use of several different NLP/querying techniques to characterize relationships between attack classes in CAPEC, CWE, and CVE; (3) development of the Security Analyst Dashboard. 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. The Dashboard is designed to work within CYBOK (though CYBOK may be used independently of the dashboard); for example, the dashboard uses the automated recommender system that underpins CYBOK to provide analysts with the capability to directly query specific entries in CAPEC, CVE, and CWE.

1 Introduction

This report describes a 12-month research activity with the principal objective of continuing development, testing and evaluation of a methodology and supporting suite of model-based engineering tools for functional risk assessment and design of cyber resilient systems. Research tasks were structured to extend the methods and support tools for the decision problem of selecting defense and resilience methods in the design and modification of cyber-physical systems. Research reported here continues the efforts of previous SERC projects, notably RT-156 and RT-172, and leverages and contributes to contemporaneous work in RT-191. The project was carried out as part of an ongoing research partnership between the University of Virginia (UVA) and Virginia Commonwealth University (VCU). The UVA team led development of methods and tools to model the consequences of cyber attacks on cyber-physical systems, and the VCU team led development of tools that relate consequences to likely attacks.

1.1 Context of Project

The University of Virginia (UVA) has been leading a research effort in System Aware Cybersecurity that includes techniques for attacking cyber-physical systems, sentinel based concepts for cyber resiliency, and tools for the selection of resilient architectures. The previous effort in this series of research, RT-172, focused on the development and selection of 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 under RT-172 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.

RT-172 made significant progress on developing decision support tools for architectural design of cyber-attack resilience. 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 Security Analyst Dashboard tools. The V1 Parser is a MagicDraw plugin that utilizes the OpenAPI to automatically extract Internal Block Diagram (IBD), Block Definition Diagrams (BDD), and Requirements structures to GraphML. The tool includes a modeling methodology that ensures the SysML blocks have a sufficient set of attributes for performing exploit chain queries.

RT-172 developed both the methodology and associated toolset with the explicit intention of generality and broad applicability. The project included development of a first prototype of a hardware/software emulation weapon system created for testing the decision-support tools. RT-191 focused on enriching and extending this test environment emulating an intelligent munition system. The emulation included features inspired by actual weapon systems as well as expanded set of situational awareness subsystems that allowed for exploration of more complex operational scenarios and attack spaces, including system-of-systems operations and attacks. In RT-191, the toolsets and methodology from RT-156, which include a hierarchical modeling approach through a War Room exercise, were used to derive mission-level requirements. 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.

1.2 Tasks

The principal objectives of RT-196 were to (1) complete development, testing and evaluation of a next-generation methodology and supporting suite of tools for assessing the vulnerability of cyber-physical systems and (2) to continue the ongoing activities to extend the methods and capabilities for vulnerability assessment to provide support for the decision problem of selecting defense and resilience methods in system design and modification, as well as support for operational decisions associated with resilience and defense. The primary tasks are as follows.

1.2.1 Architectural Selection Methodology and Tools

The objective in this task was to complete the development of the architectural selection methodology and tools begun in RT-172, referred to as V1, and develop a new generation, V2. RT-191 was initiated to evaluate the initial version of a usable tool set (V0), with evaluation results completed July 2018. RT-172 advanced V0 to a more advanced support capability (V1) that was finalized as part of RT-196. The RT-196 also included adding additional tool capabilities to address more complex system configurations with enhancements to allow users to significantly increase their productivity.

1.2.2 Support War Room Activities

This task centered on providing support for RT-191 war rooming activities by participating in scenario development, War Room Blue and Red live sessions, and consequent development of systems models. 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.

1.2.3 Security Analyst Dashboard

The objective of this task was to develop new concepts and prototype for a *Security Analyst Dashboard* to support decisions about where to add sentinels and other resilience and defense mechanisms. The V2 architectural selection methodology and tools provide an efficient way to evaluate the threats and vulnerabilities of a given system. However, they do not provide explicit support for the decision of how to modify the defense and resilience architecture to improve overall system resiliency. To address this decision problem, research focused on: development of scoring algorithms that attempt to provide the analyst with an understanding of the interactions between consequence (from system requirements modeling) and likelihood (from analysis of historical attacks).

1.2.4 Sentinel Vulnerability Assessment Methodology & Algorithms

The objective in this task was to develop new concepts for self-securing systems by dynamically adjusting Sentinel vulnerability assessment algorithms, based on related sentinel alarms that occur or other operational and functional data. This work leverages the machinery from the Security Analyst Dashboard also topic modeling and other natural language processing algorithms.

2 Background

2.1 Cyber Security Requirements Methodology (CSRM)

This project builds upon some of the techniques and methods RT-191, which was led by UVA with the participation of the Software Engineering Institute (SEI) and the US Army's Armament Research Development and Engineering Center (ARDEC). A principal outcome of RT-191 was the specification of the Cyber Security Requirements Methodology (CSRM) and testing on an emulated concept-stage weapons system. CSRM is a methodology to develop cyber security requirements during the preliminary design phase for physical systems [3]. The methodology addresses the integration of both defense and resilience solutions and security-related software engineering solutions. CSRM consists of six steps:

1. High-level development of functional and architectural system descriptions by a systems engineering (SE) team using tools such as SysML
2. Blue team consequence elicitation and analysis, whose deliverable is a prioritized list of undesirable functional outcomes
3. SE team derivation of potential resilience solutions based on the results of step 2
4. Red team prioritization of defense, resilience, and software engineering solutions
5. SE team refactoring of system descriptions based on Red team recommendations
6. Blue team response to the refactored system descriptions.

In RT-191, a hypothetical, concept-stage weapon system, known as Silverfish, was used to demonstrate the CSRM process. Silverfish consisted of a rapidly deployable set of approximately 50 ground-based munition systems, termed obstacles. These obstacles deny a geographic area from unauthorized trespassers through the use of force, if necessary, to support the protection of a strategically sensitive location. An operator remotely monitors this denied area using a variety of sensors and visual surveillance. The operator controls the arming, disarming, and firing of the obstacles remotely via a wireless communication network. The final recommendations of the CSRM exercise regarding resiliency, in order of priority, involved adding diverse communication systems, adding resilient design patterns to the situational awareness components of the system, and adding resilient design patterns to the system's weapon control components. These results are used for comparison with the recommendations of the tools described in Section 4.1.

3 Selecting System Specific Cybersecurity Attack Patterns Using Natural Language Processing

In a previous research project, the research team investigates the use of topic modeling [4] on the Common Attack Pattern Enumeration and Classification (CAPEC) database of historical cybersecurity attacks[1] [5]. The results of that research were preliminary and demonstrated that topic modeling could be used to extract information from the CAPEC database. In RT-196, we demonstrated how a natural language processing (NLP) technique called topic modeling could be used to match entries in CAPEC with a system. 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 is measured using the Kullback-Leibler divergence [6]. The work in this section was presented at *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* and published in the proceedings [7].

3.1 Background

The CAPEC database is composed of 512 patterns that represent common attacks on software and computer systems. Each attack pattern has a text description with common fields including a summary of the attack, attack prerequisites, and links to related attack patterns. CAPEC was created to provide a publicly accessible repository for historical information on attacks that would give cybersecurity researchers and professionals the capability to learn from experience.

The development of systems to automatically retrieving attack patterns from CAPEC has been the focus of other research projects. Yuan *et al.* [8] develop a software tool for retrieving attack patterns. This method maps CAPEC attack patterns to Microsoft STRIDE categories but relies on user input such as knowledge of the attacker's skill level. Kotenko and Doynikova [9] propose a technique for generating random attack sequences utilizing the CAPEC database but requires knowledge about the attacker. The proposed method outlined in this section is a more general concept that relies solely on the text provided for each attack pattern in the CAPEC database. However, the work outlined previously on retrieving attack patterns from CAPEC and generating sequences of random attacks could be used in conjunction with proposed NLP method for matching attack patterns to a system description.

Topic modeling has been utilized in several domains and for several applications. It is most predominantly used for document clustering and classification [4], but it has also been used for document

[1]https://capec.mitre.org/index.html

retrieval [10], software traceability [11], sentiment analysis [12], and sentence ordering [13]. Topic modeling has also been applied to cybersecurity problems. Aswani *et al.* [14] use a topic model to extract information about SSH logs with the goal of classifying legitimate users from brute-force attackers. Kolini and Janczewski [15] used topic modeling to identify clusters and topics of national cybersecurity strategies. Temporal trends in CVE were analyzed using topic modeling by Neuhaus and Zimmermann [16].

3.2 Topic Modeling

The following description of topic modeling was previously included in RT-172 report [17] but is reproduced here for convenience.

Topic modeling is a machine learning technique commonly used in NLP that estimates latent or hidden topics from a corpus of documents. One method for describing a document commonly used in NLP is to count the number of times each word appears in the text and store these counts in a vector. This method is often called "bag-of-words". We will represent the length of this bag-of-words vector using L. For corpora with a large number of documents and documents containing a large amount of text, this vector can quickly grow which leads to numerous problems when attempting to perform analytics, such as classificaiton or topic modeling. A topic model represents each document in the corpus as a topic distribution with T topics. The number of topic distributions is generally chosen so that $T << L$ so that the topic distributions can be used in place of the bag-of-word vectors when performing analysis.

There are several types of topic models including hierarchical topic models [18], correlated topic models [19], and supervised topic models [20], but we limit our description of the method to the basic latent Dirichlet allocation (LDA) [4]. 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. The presence of a word in a document is used instead of the word count. Let **v** be a vector of binary variables where $v_i = 1$ indicates that the i^{th} word appears in the document, and $v_i = 0$ indicates that 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** $= (w_1, \ldots, w_N)$. The corpus is composed of M documents and represented by $\mathcal{D} = \{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$.

The following generative process is assumed for each document in the corpus when using the basic LDA formulation:

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 $P(w_n|z_n, \beta)$

It is assumed that the number of topics T is known and fixed in this process for generating each document,. The joint probability 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:

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

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

$$P(\mathbf{w}|\alpha, \beta) = \int P(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{\mathbf{z}} P(z_n|\theta) P(w_n|z_n, \beta) \right) d\theta. \tag{2}$$

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

$$\begin{aligned} P(\mathcal{D}|\alpha, \beta) = \sum_{m=1}^{M} \Bigg[\int & P(\theta_m|\alpha) \\ & \left(\prod_{n=1}^{N_m} \sum_{\mathbf{z}_m} P(z_{mn}|\theta_m) P(w_{mn}|z_{mn}, \beta) \right) d\theta \Bigg]. \end{aligned} \tag{3}$$

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:

$$P(\mathbf{z}, \theta|\mathbf{w}, \alpha, \beta) = \frac{P(\mathbf{z}, \mathbf{w}, \theta|\alpha, \beta)}{P(\mathbf{w}|\alpha, \beta)}. \tag{4}$$

The posterior distribution is intractable for an exact solution but other estimation methods, such as variational inference [21] and Gibbs sampling [22], can be employed to estimate these variables and parameters.

3.3 Methodology

This section outlines our proposed methodology for selecting attack patterns that could be used to attack a system using NLP. The proposed methodology aims to find attack patterns in CAPEC that are "close" to the system description in the topic space. The methodology returns a ranked list of attacks and is intended to be used as a suggestion for cybersecurity experts when assessing the vulnerabilities of a system.

The steps of the methodology are as follows:

1. Extract and process the text of the attack database.
2. Learn a topic model of the attack database.

3. Extract the text describing the system from a model or other documents relevant to the system.
4. Create a term-frequency vector from the extracted text of the system using only words that match those in the attack database vocabulary.
5. Estimate the topic posterior distribution of the system using the attack database topic model and the term-frequency vector of the system.
6. Calculate the KL divergence between the topic distribution of the system and the topic distribution of each attack.
7. Rank attacks using the KL divergence measure from minimum to maximum.

The proposed methodology is heavily reliant on the text that describes the system. In some cases, design documents or operations manuals could be used in place of the system description or in addition to the system description. If this documentation is not available, a model of the system could be constructed. The text can be extracted from this model. In our example, documentation of the system was not available and the Systems Modeling Language (SysML) [23] was used to construct a model.

The KL divergence is a common measure for evaluating the similarity of two distributions:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}, \tag{5}$$

where P and Q are two discrete probability distributions. If P and Q match exactly, then $D_{KL}(P||Q) = 0$. Other measures of distribution similarity could be used in place of the KL divergence, but this measure was selected due to its widespread use in information theory.

3.4 Application of Approach

In this section, we describe our work on the application of the proposed methodology to an early prototype design of Silverfish which has less functionality and capability than the full Silverfish system. The text used in this example application was extracted from a SysML model of the system.

To the best of our knowledge, there are no methods for selecting an attack pattern for a given system using text describing both the attacks and the system. The two methods described earlier in the paper ([8] and [9]) require prior knowledge about the skill level of the attacker. Therefore, these methods should not be compared to the proposed method. We limit our analysis to processing the text from CAPEC, comparing different approaches to estimating the parameters of the LDA and the number of topics, and testing the proposed methodology.

The first two steps of the proposed methodology extract text from the CAPEC database and estimate a topic model. Topic modeling is an unsupervised learning algorithm which makes it difficult to tune model parameters and validate the model. There are numerous decisions that must be made when constructing a topic model ranging from how to process the data to the type of topic model, e.g. LDA or correlated, to the number of topics. There is no standard methodology to addressing these decisions. Further, due to the unsupervised nature of the problem, metrics for evaluating the quality of the model are not standardized.

The text data must be extracted and processed before a topic model can be learned. The first step for processing the CAPEC text is to remove any attack patterns that do not have a description. This reduces the number of attack patterns from 512 to 500. In its unprocessed form, the CAPEC database has a vocabulary of over 11,000 terms. For a normal corpus, this would be a reasonable or even small number of terms. However, the length of each attack description is relatively short when compared with documents usually analyzed in NLP. Further, the unprocessed form of the text includes punctuation, numbers, and repeated terms due to capitalization. In an effort reduce the size of the vocabulary and to only give a more concise representation of the text in the database, the following common NLP processing steps are performed on the corpus:

1. Remove punctuation
2. Convert all upper case letters to lower case
3. Remove numbers
4. Remove *stop words*
5. Perform stemming

Several words that commonly occur in the English language are not relevant to NLP and can inject noise into the modeling, i.e. “the”, “and”, “a”, “or”, etc. It is common to remove these *stop words* before performing NLP. Stemming the terms in a vocabulary is also common practice. For example, “attack” and “attacks” are considered separate terms in the raw text, but performing word stemming reduces them to a single term. Porter’s stemming algorithm [24] was used as the stemming procedure.

Once these processing steps are performed, the vocabulary is reduced to 4274 unique terms. However, there are still several terms in the vocabulary that appear only a handful of times, therefore, terms that appear in less than one percent of the documents are removed. This reduces the vocabulary to 1307 terms and concludes our processing procedure. The term frequency is calculated for each term in the processed corpus and stored in a document-term matrix.

The next step of the methodology is to learn a topic model from the processed CAPEC text. Choosing the number of topics for a topic model can be difficult as there is no “correct” answer due to the unsupervised nature of the problem. Further, the parameters of the standard LDA can be estimated using either variational estimation or Gibbs sampling. We combine these two problems by evaluating topic models estimated using both methods with the number of topics ranging from 2 to 50. Three topic modeling evaluation metrics are used: CaoJuan2009 [25], Arun2010 [26], and Deveaud2014 [27]. The objective is to find a topic model that minimize the first two metrics and maximize the third. Figures 1 and 2 display these evaluation metrics for variational estimation and Gibbs sampling, respectively. The topic models using variational estimation improve as the number of topics is increased. However, the Deveaud2014 metric increases around 5 topics when using Gibbs sampling and stays relatively constant as the number of topics increases. The other two metrics improve as the number of topics is increased. This relationship indicates that

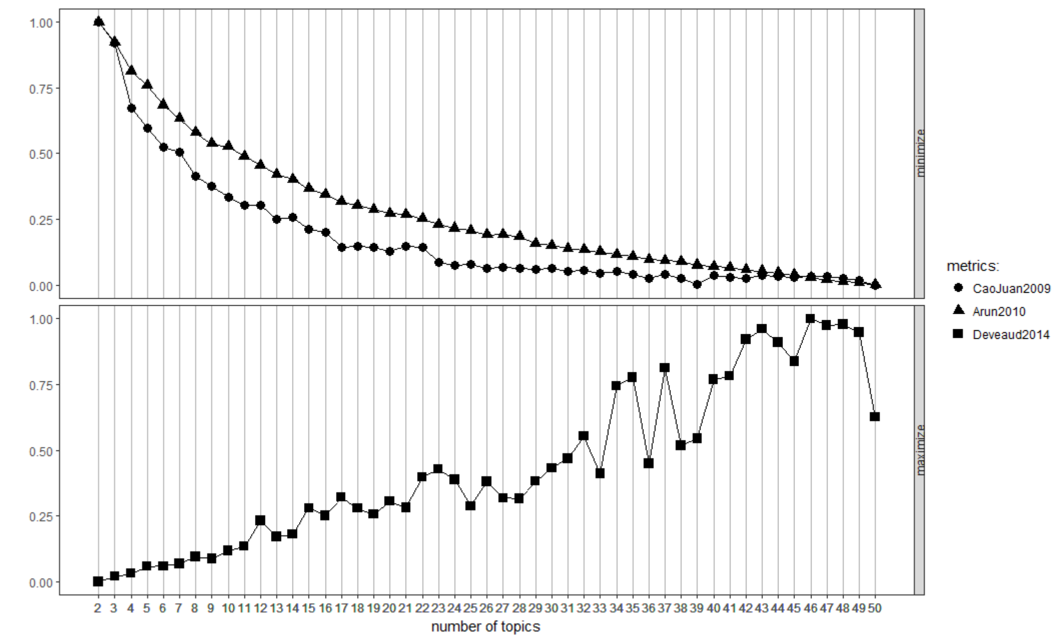


Figure 1: Evaluation metrics for variational estimation.

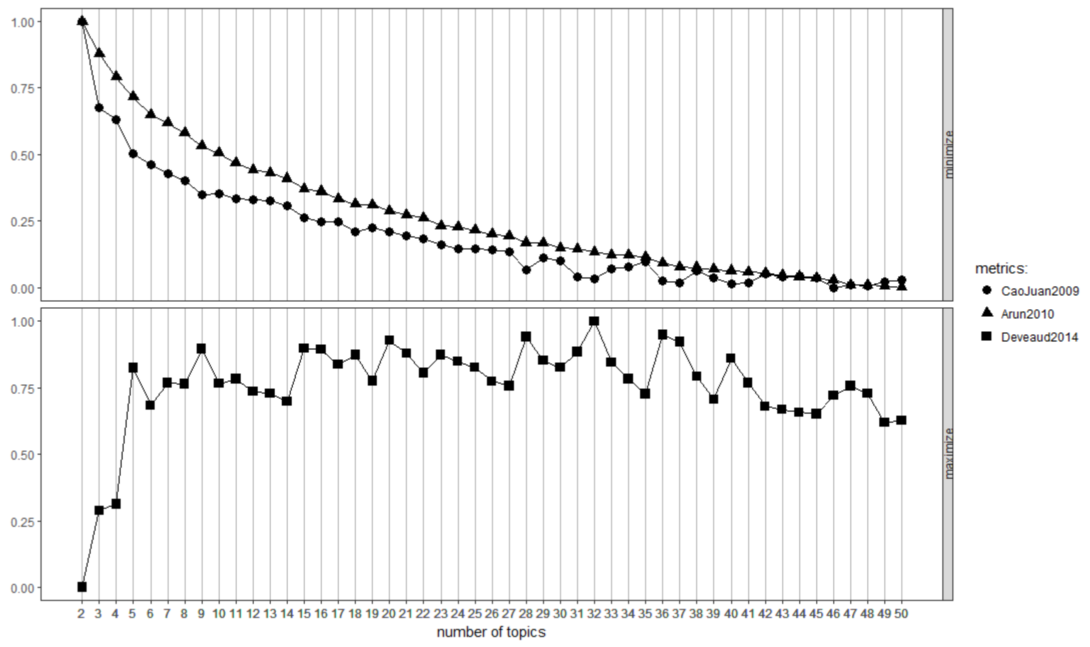


Figure 2: Evaluation metrics for Gibbs sampling.

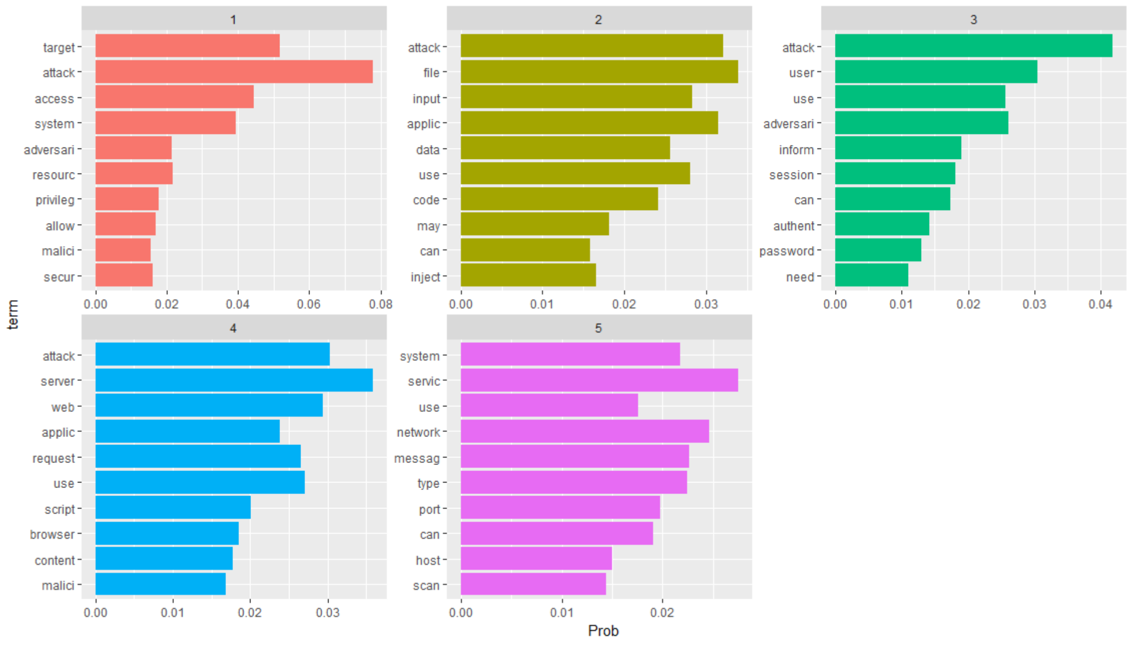


Figure 3: Top 10 words per topic.

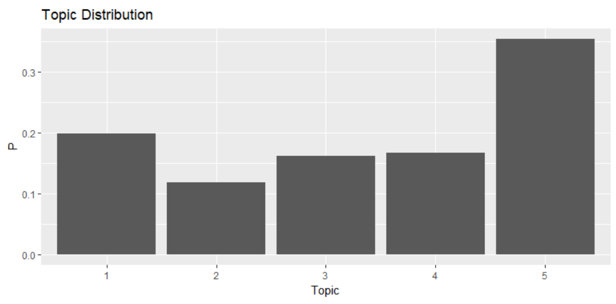


Figure 4: Posterior topic distribution for example system.

we can either choose a large number of topics and the variational method or we can select a small number of topics and use the Gibbs sampling estimation method. In order for topics to be more interpretable, we decide to the Gibbs sampling procedure to learn a topic model with 5 topics for estimating the parameters of the model.

Gibbs sampling is used to estimate a standard LDA topic model for the CAPEC database. In order to remove the effects of random initialization of parameters, five models (each with a different seed) are estimated and the model with the maximum posterior likelihood is selected. The top 10 words for each topic are displayed in Figure 3.

The next step is to collect text that describes the system. In this application of the methodology, text was extracted from the SysML model. The text was processed using the same processing procedure as the CAPEC text. However, removing stop words and filtering based on sparsity are not necessary because only terms that appear in the CAPEC vocabulary are used for the system term vector. Term frequencies were calculated for words that are in the CAPEC vocabulary and assigned to word vector $\mathbf{w}^s$. The posterior distribution of topics for the system $\mathbf{z}^s$ is estimated using $\mathbf{w}^s$ and the learned topic model from the CAPEC database. Figure 4 displays the posterior topic distribution for the system.

Table 1: Top 5 Attacks. This table contains the CAPEC ID, the distance between the attack topic distribution and the system topic distribution, the title of the attack, and the summary of the attack as listed in CAPEC.

CAPEC ID	Distance	Title	Summary
619	0.001	Signal Strength Tracking	In this attack scenario, the attacker passively monitors the signal strength of the target's cellular RF signal or WiFi RF signal and uses the strength of the signal (with directional antennas and/or from multiple listening points at once) to identify the source location of the signal. Obtaining the signal of the target can be accomplished through multiple techniques such as through Cellular Broadcast Message Request or through the use of IMSI Tracking or WiFi MAC Address Tracking.
615	0.003	Evil Twin Wi-Fi Attack	Adversaries install Wi-Fi equipment that acts as a legitimate Wi-Fi network access point. When a device connects to this access point, Wi-Fi data traffic is intercepted, captured, and analyzed. This also allows the adversary to act as a "man-in-the-middle" for all communications.
495	0.007	UDP Fragmentation	An attacker may execute a UDP Fragmentation attack against a target server in an attempt to consume resources such as bandwidth and CPU. IP fragmentation occurs when an IP datagram is larger than the MTU of the route the datagram has to traverse. Typically the attacker will use large UDP packets over 1500 bytes of data which forces fragmentation as ethernet MTU is 1500 bytes. This attack is a variation on a typical UDP flood but it enables more network bandwidth to be consumed with fewer packets. Additionally it has the potential to consume server CPU resources and fill memory buffers associated with the processing and reassembling of fragmented packets.
623	0.008	Compromising Emanations Attack	Compromising Emanations (CE) are defined as unintentional signals which an attacker may intercept and analyze to disclose the information processed by the targeted equipment. Commercial mobile devices and retransmission devices have displays, buttons, microchips, and radios that emit mechanical emissions in the form of sound or vibrations. Capturing these emissions can help an adversary understand what the device is doing.
603	0.009	Blockage	An adversary blocks the delivery of an important system resource causing the system to fail or stop working.

The distance between the system topic distribution and the topic distribution for each attack in the CAPEC database was measured using the KL Divergence. The five closest attacks, in terms of distance in the topic space, found using this method are displayed in Table 1. The table also includes the distance between the system topic distribution and the attack topic distribution and the summary of the attack as listed in the CAPEC database. The proposed method is difficult to validate because ranking the best attacks is a subjective task. However, the returned results demonstrate that the selected attacks target the communication subsystem. While the communication system is encrypted, the Compromising Emanations Attack describes an attack where the signals are monitored. The strength of the signal or the frequency of transmission could be used by an adversary to gain knowledge about the system. This type of attack would not allow the adversary to gain access to the system but it could be used to degrade the effectiveness of the system. In a separate activity, a group of cyber analysts conducted a Red Team activity on the prototype system and selected similar types of attack patterns as the proposed NLP method.

The five attacks with the greatest distance between the system topic distribution and the attack topic distribution as suggested by the proposed method are displayed Table 2. This offers another form of validation for the method. The attacks that are farthest from the system topic distribution generally target keywords, browsers, and web applications. As the system does not have access to the internet and is essentially a closed system, it seems reasonable that these attacks should not be considered or be considered as low-likelihood of occurring.

We would like to point out that this list should be used as suggestion of possible attacks and not

Table 2: Bottom 5 Attacks. This table contains the CAPEC ID, the distance between the attack topic distribution and the system topic distribution, the title of the attack, and the summary of the attack as listed in CAPEC.

CAPEC ID	Distance	Title	Summary
199	1.03	XSS Using Alternate Syntax	An adversary uses alternate forms of keywords or commands that result in the same action as the primary form but which may not be caught by filters. For example, many keywords are processed in a case insensitive manner. If the site's web filtering algorithm does not convert all tags into a consistent case before the comparison with forbidden keywords it is possible to bypass filters (e.g., incomplete black lists) by using an alternate case structure. For example, the "script" tag using the alternate forms of "Script" or "ScRiPt" may bypass filters where "script" is the only form tested. Other variants using different syntax representations are also possible as well as using pollution meta-characters or entities that are eventually ignored by the rendering engine. The attack can result in the execution of otherwise prohibited functionality.
244	1.02	XSS Targeting URI Placeholders	An attack of this type exploits the ability of most browsers to interpret "data", "javascript" or other URI schemes as client-side executable content placeholders. This attack consists of passing a malicious URI in an anchor tag HREF attribute or any other similar attributes in other HTML tags. Such malicious URI contains, for example, a base64 encoded HTML content with an embedded cross-site scripting payload. The attack is executed when the browser interprets the malicious content i.e., for example, when the victim clicks on the malicious link.
32	1.01	XSS Through HTTP Query Strings	An adversary embeds malicious script code in the parameters of an HTTP query string and convinces a victim to submit the HTTP request that contains the query string to a vulnerable web application. The web application then procedes to use the values parameters without properly validation them first and generates the HTML code that will be executed by the victim's browser.
86	1.00	XSS Through HTTP Headers	An adversary exploits web applications that generate web content, such as links in a HTML page, based on unvalidated or improperly validated data submitted by other actors. XSS in HTTP Headers attacks target the HTTP headers which are hidden from most users and may not be validated by web applications.
63	0.91	Cross-Site Scripting (XSS)	An adversary embeds malicious scripts in content that will be served to web browsers. The goal of the attack is for the target software, the client-side browser, to execute the script with the users' privilege level. An attack of this type exploits a programs' vulnerabilities that are brought on by allowing remote hosts to execute code and scripts. Web browsers, for example, have some simple security controls in place, but if a remote attacker is allowed to execute scripts (through injecting them in to user-generated content like bulletin boards) then these controls may be bypassed. Further, these attacks are very difficult for an end user to detect.

a definitive ranking of the most harmful attacks. It should be used in conjunction with domain knowledge when developing a cybersecurity system.

3.4.1 Future Work

There are several avenues for possible future work. First, the method is only applied to a single relatively simple system and to one SysML model of that system. A more rigorous study in the future should include several types of systems and a combination of text extracted from SysML models and existing documentation. Second, improved topic models could lead to a better list of attack patterns. While the standard LDA was used in this study, numerous versions of topic models exist. One limitation of the CAPEC database is the relatively short description of each attack. Several methods have been proposed for short texts that include using auxiliary texts such as wikipedia [28, 29]. Domain knowledge of a subject can also be leveraged to improve the performance of topic models [30, 31]. Finally, the method was demonstrated for the CAPEC database but it could be expanded to CVE, CWE, or any other cybersecurity database with text.

CSRM Step	STRAT Tool Support
1. High level, tool-based, system description produced by the SE team, including the basic system architecture and functional description in SysML	Mission and System Specification
2. Blue team operational risk assessment, whose deliverable is a prioritized list of undesirable functional outcomes, and consequence analysis based on the system description	Systems-theoretic consequence analysis
3. SE team derivation of potential resilience solutions based on the results of operational risk assessment	Model-based solution identification
4. Red team prioritization of defense, resilience, and software engineering solutions	
5. SE team refactoring of system descriptions based on Red team recommendations	Simulation-based solution evaluation
6. Blue team response to the refactored system descriptions	

Table 3: CSRM process and associated tool support from STRAT. Overlap represents the steps of CSRM that the respective component of STRAT supports.

4 Tools Effort

The tools efforts follow two broad themes: (1) general tool support for conducting CSRM in section 4.1 and (2) more specific tools for conducting cyber analysis in section 4.2.

4.1 Systems-Theoretic Resiliency Assessment Tool (STRAT)

This section describes the tools used to support CSRM and identify appropriate resiliency solutions based on systems-theoretic control and behavior models. The methodology expands on the concepts defined in CSRM that lead to the identification of potential resiliency-enhancing strategies for a given system. CSRM identifies potential resiliency solutions based on the mission and system descriptions, inputs from stakeholders, and the judgment of the Systems Engineering team. The methodology introduced in this section can be used to augment the CSRM by providing model-based justification for the Systems Engineering (SE) team, or the methodology can be used on its own to identify and evaluate appropriate resiliency enhancements. The tools and models described in this section, the Systems-Theoretic Resiliency Assessment Tool (STRAT) is composed of four main components: mission and system specification, systems theoretic consequence analysis, model-based solution identification, and solution evaluation. The components of STRAT and how they support the broader systems engineering methodology of CSRM are shown in Table 3.

4.1.1 Development of Mission and System Specifications

STRAT method shares its initial steps with CSRM. Both the system and the mission it performs are specified at a high-level along with a, preferably rank-ordered, set of unacceptable outcomes to that mission. These pieces of information form the basis of the STAMP-based analysis from which the system's control structure and potential loss scenarios are derived. Ideally, the mission and system descriptions are generated by consensus in an iterative process between the SE team and the system owners. However, if the system owners are not available for engagement or if the SE team represent the system owners, then the SE team can complete the descriptions independently. At a minimum, the initial mission description should describe in natural language:

1. The overall mission objective and any sub-objectives,
2. The greater purpose the mission supports,
3. Criteria for mission success and failure,
4. and any constraints on the environment in which the system operates to complete the mission.

The system description shall also describe in natural language how the system is intended to complete the mission, any known components within the system, a basic functional description of the system's operation, and any other known constraints on the system's operation. Preliminary mission and system descriptions should be developed by the SE team and the system owners over two or three iterations and the length of the descriptions should not exceed one to two typed pages. Agreeing upon a concise description has the dual benefit of scoping analysis to a more manageable degree for complex systems as well as preventing confusion about the goals of the mission and how the system is used to help reach those goals.

Following the development of the mission and system descriptions, if the system owners are available for engagement, the STRAT method borrows from Step 2 of the CSRM- the Blue Team consequence elicitation meeting. The Blue Team meeting engages the SE team with the system owners to elicit a prioritized set of undesirable consequences or outcomes with respect to the use of the system in the mission. The development of the list of undesirable outcomes is based on the agreed upon mission and system specifications described previously. The SE team is responsible for facilitating the discussion and documenting the outcomes along with other relevant pieces of information from the system owners. Such information could include, but is not limited to, the components that would likely need to be attacked to produce that outcome and the potential method of attack. The CSRM, for example, identifies an additional piece of information- STAMP type- to further characterize the undesirable outcome in terms of the control action (or lack thereof) needed to produce the outcome. All of this information collected from the systems owners forms the foundation for the STAMP-based analysis and construction of the system's control model.

In the event that conducting a Blue Team meeting as described in the CSRM is not possible, then the SE team will need to rely on their understanding of the system and mission description and personal expertise. Under these circumstances, the value of having a clear and consistent system and mission description becomes evident. If the descriptions are easily understood, then it becomes more likely that a non-user or non-expert will be able to identify valid, and well-formed, undesirable outcomes. Regardless of the team that develops the list of undesirable outcomes, the rationale behind each outcome should be documented to enable any cascading effects in future analysis to be traceable.

4.1.2 Systems-Theoretic Consequence Analysis

Following the specification of the mission, system, and undesirable outcomes, the SE team performs a systems-theoretic consequence analysis to define the system's functional control structure, behavior, and potential scenarios that might produce undesirable outcomes. More specifically, this step of the methodology is based on Leveson's STAMP model and STPA/STPA-Sec analysis tools. The STRAT follows the concepts of the STAMP model and performs most of the steps in the STPA-Sec analysis tool, but the goals of each method differ. STPA-Sec identifies scenarios, that could be the result of a cyber-attack, to focus cybersecurity efforts; however, STRAT uses the STAMP-based analysis to guide the construction of models that are used to identify appropriate locations and types of cyber-resilience strategies [32].

STPA and STPA-Sec begin with the identification of unacceptable losses in the mission at hand. STRAT uses the information collection methods described in section 4.1.1 to perform this same task. The set of undesirable outcomes generated by the SE team or the system owners are directly mapped into the unacceptable losses used in the consequence analysis. Unacceptable losses in STPA-Sec syntax are high-level events that typically imply total mission failure. Consequently, it may be possible that some of the undesirable outcomes generated in the previous step may be too specifically defined to be well-formed unacceptable losses. In such cases, there is likely an implicit higher-level loss event tied to that outcome that should be defined. For example, multiple undesirable outcomes may be able to be categorized as a more general type of unacceptable loss. It should be noted, however, that the one of the purposes of beginning with the definition of unacceptable losses is to scope later analysis, and therefore, the set of unacceptable losses should not be so specific that the problem space becomes too complex.

After the definition of unacceptable losses in the mission, a set of hazardous conditions that could contribute to one of the unacceptable losses are identified. In fact, some of the more specific undesirable outcomes from the Blue Team elicitation are likely to describe a hazardous scenario that could lead to a higher-level loss event. Hazardous conditions outline scenarios that could occur during the operation of the system within the mission that would lead to an unacceptable loss if they were to occur in combination with the presence of a worst-case environment. Young and Leveson illustrate this by describing a nuclear power plant that has an unacceptable loss of not producing power to the grid. A hazardous scenario for the power plant would be the shutdown of the reactor, however, the associated unacceptable loss only occurs if there are no auxiliary generators or if the reactor is shutdown longer than the endurance of the auxiliary generators [32]. Following the identification of hazardous scenarios, the basic control structure of the system is defined. The development of the control structure is based on the controller, actuator, controlled process, and sensor feedback loop seen in Figure 5.

The system's control structure emerges as these loops are stacked on top of one another, in parallel, or merged together, similar to the control structure of a fictional missile defense system shown

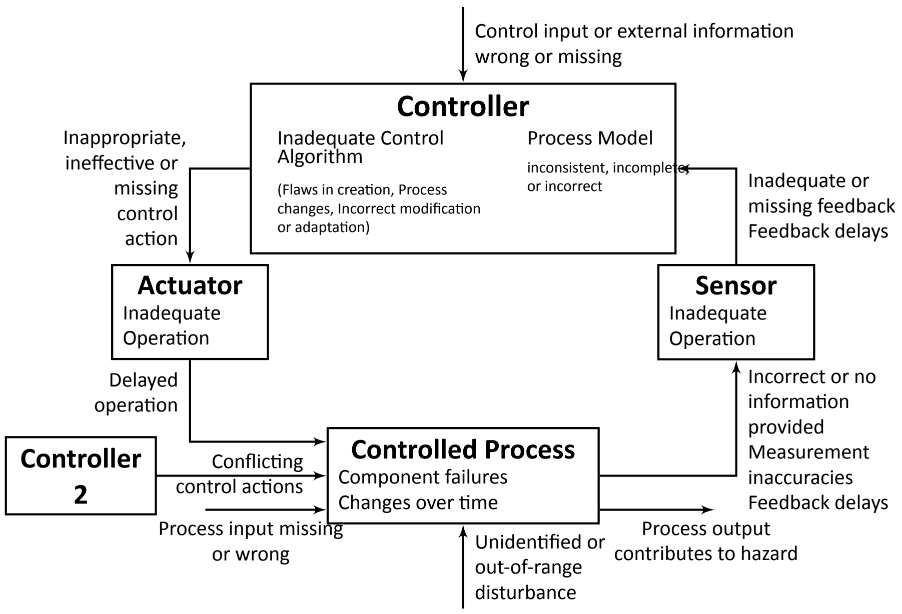

Control input or external information wrong or missing
Controller
Inadequate Control Algorithm
(Flaws in creation, Process changes, Incorrect modification or adaptation)
Process Model
inconsistent, incomplete, or incorrect
Inappropriate, ineffective or missing control action
Inadequate or missing feedback Feedback delays
Actuator
Inadequate Operation
Sensor
Inadequate Operation
Delayed operation
Incorrect or no information provided Measurement inaccuracies Feedback delays
Controller 2
Conflicting control actions
Controlled Process
Component failures
Changes over time
Process input missing or wrong
Unidentified or out-of-range disturbance
Process output contributes to hazard

Figure 5: The generic control loop structure that is used to formulate the control model. (adapted from [1]).

in Figure 6. The combination of these loops creates a hierarchy of controllers and controlled processes that begins to describe the technological and organizational mechanisms that the system uses to operate within its mission domain. More specifically, the hierarchical control structure defines how commands and control actions propagate from the higher-level controllers to lower-level controllers or controlled processes and how those lower-level entities provide feedback to their higher-level controllers [1]. Identifying how the system accomplishes these tasks is the first step to understanding how unintended or uncontrolled system behavior can lead to unacceptable losses.

Defining the control structure allows for the enumeration of the control actions available to each controller within the hierarchy. Since the STAMP causality model asserts that hazardous conditions are the result of performing control actions improperly, the enumeration of control actions allows the SE team to identify the scenarios under which improperly implemented control actions lead to hazardous conditions, and thus, potential unacceptable losses. Improper control actions can be categorized into four types of implementation:

1. Providing the control action leads to a hazardous condition
2. Not providing the control action leads to a hazard
3. Providing the control action too early, too late, or in the incorrect order leads to a hazard
4. Stopping a control action too soon or performing a control action too long leads to a hazard.

By creating a table of the possible control actions and how each type of improper implementation of those control actions can lead to hazardous conditions, the SE team begins to identify potential areas of concern within the control structure through the process of elimination. Some control actions will not have scenarios that create hazardous conditions for all of the improper implemen-

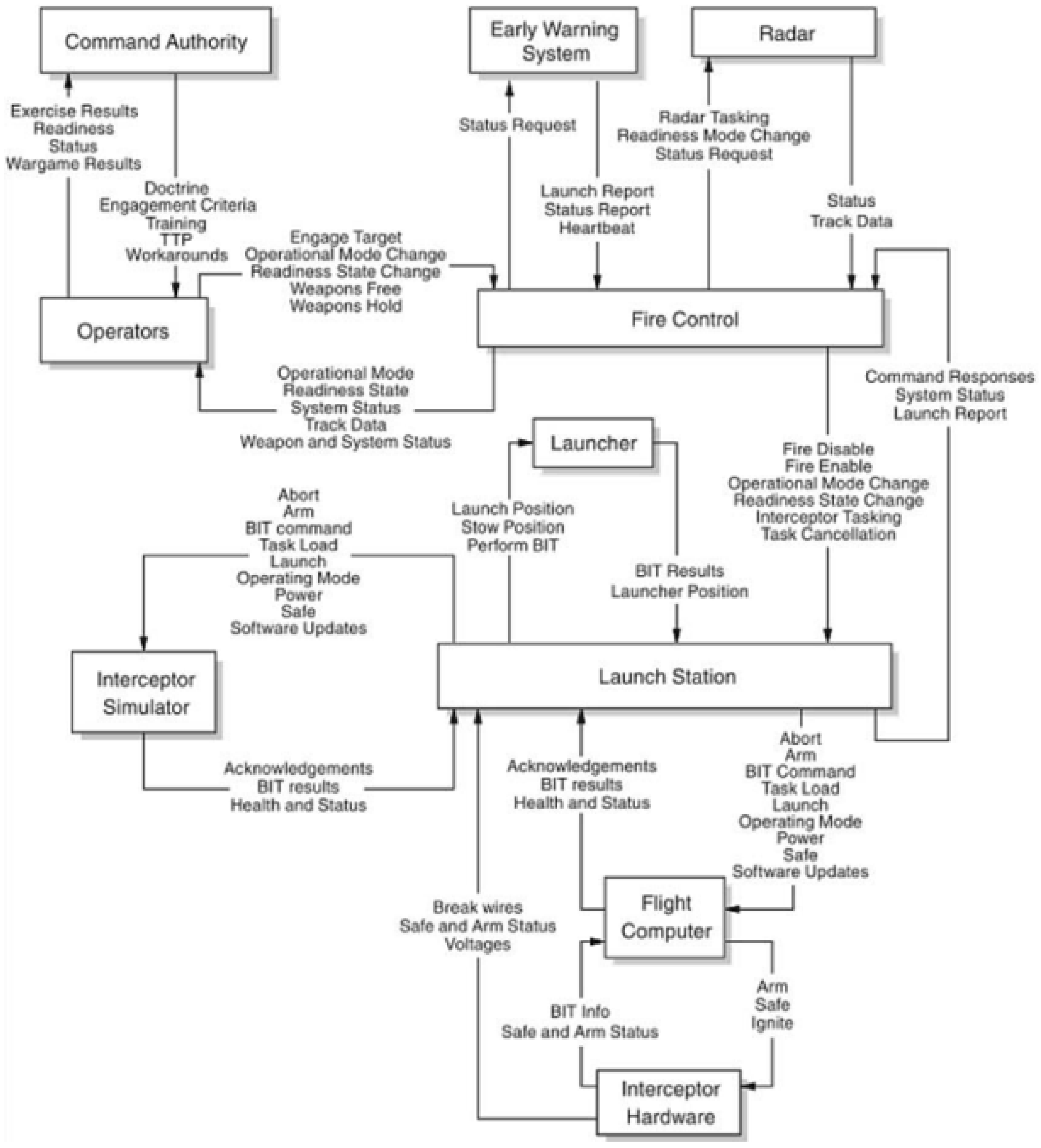


Figure 6: A “stacked” control structure of a fictional missile defense system (From [2]).

tation types because of the nature of the control action. Thus, those cases can be ignored in future analysis, thereby reducing the problem space. Furthermore, the SE team will be able to take note of any control actions that can lead to the same hazardous condition for multiple improper implementation types. These control actions can be flagged as areas to investigate more thoroughly in later analysis.

The final step in STPA and STPA-Sec involves the construction of causal scenarios that describe why an improper control action was taken. The identification of these scenarios facilitates an understanding of the impact cyber events have on the mission- something with which traditional security methodologies may struggle. Furthermore, identifying the potential mechanisms through which adverse outcomes can occur helps motivate the choice of appropriate resilient design patterns later on.

The main artifacts of the STAMP-based consequence analysis are the definition of the system's functional control structure and the documentation of the relevant system losses, hazards, hazardous control actions, and causal scenarios. These artifacts aid the SE team in understanding the manner in which vulnerabilities can propagate through the system in addition to forming the foundation for the construction of the system and behavior models.

4.1.3 Model-based Solution Identification

4.1.3.1 The System Model

While the STAMP-based consequence analysis facilitates understanding of the system's control structure and identifies potential pathways for vulnerabilities that lead to adverse outcomes, it does not produce an analyzable model. Consequently, it becomes advantageous to represent the system's control structure, unacceptable outcomes, and other STAMP-based analysis information in graphical form. This representation allows for the visualization of the control actions, the resultant changes to the system, and the emergence of mission-level consequences from those actions. Furthermore, the graphical formulation allows for the beginnings of a quantification of the qualitative subject matter obtained in the mission and system specifications and the consequence analysis.

The graphical representation of the system, its control structure, and the consequence analysis necessitates a special definition of its graphical objects. This graph, known as the specification graph, or S-graph [33], shares similarities to the definition of a multidigraph or quiver [34]. However, the S-graph's vertices and edges are supersets of dissimilar sets of vertices and edges. The need for differing types of vertices and edges arises from the representation of the elements in the control loop shown in Figure 5, and therefore STRAT modifies the general notion of S-graph into what we call the *Sim-graph*. The Sim-graph models actors in the system by having its vertices represent an entity from the generic control loop, a combination of those entities, a physical state, or a function that represents an outcome. It follows that the edges in the graph represent the actions performed by or resulting from the actors.

Following these concepts, the Sim-graph is composed of a combination of six types of vertices: outcome vertices, state vertices, actuator vertices, sensor vertices, controller vertices, and meta vertices. The outcome vertices describe the presence or absence of certain conditions from the consequence analysis, such as the presence of a hazardous condition or the occurrence of an unacceptable loss. State vertices are broadly defined as the set of variables or controlled processes that are not also controlling a lower-level process, such as vehicle's location, speed, etc. As the name implies, actuator vertices represent an actuator in the system's control structure that receives input from a controller and acts upon a controlled process. Likewise, sensor vertices represent a sensor in the system's control structure that monitors a controlled process or state and sends feedback to a controller. The Sim-graph's controller vertices represent a controller in the system's hierarchical control structure. Due to the control hierarchy, a controller vertex will both receive inputs from a higher-level actuator and send control actions to a lower-level actuator, and vice-versa for its corresponding sensor vertices, unless the vertex is the highest-level controller in the system. Finally, meta vertices can be used to represent a combination of controllers, actuators, sensors, or controlled processes. This allows for the possibility that an entity in the hierarchical control structure shares the responsibilities of two or more of the parts of the generic control loop. An example of such shared responsibilities is illustrated in Figure 6.

The edges of the Sim-graph are also of different types. These types include action edges, feedback edges, and conditional edges. Action edges represent the control actions or dynamics through which a higher-level vertex influences a lower-level vertex. For example, a controller vertex may have multiple action edges from itself to the subsequent actuator vertex that represent the control actions available to that controller in the system's control structure. Feedback edges represent data or information that is propagated from a lower-level vertex to a higher-level vertex, such as the feedback from a sensor to its higher-level controller. Finally, the conditional edges represent the inputs to an outcome vertex.

Using these definitions to the varying types of vertices and edges, a mathematical definition of the Sim-graph is as follows. The specification graph, S, is a 4-tuple similar to a multidigraph, or quiver:

$$S := (V, E, p, t)$$

where V is the superset of nodes in the graph, E is the superset of edges, $p : E \to V$ assigns each edge to its parent vertex, and $t : E \to V$ assigns each edge's target vertex. Furthermore, the superset V is defined:

$$V \supseteq O, X, A, D, C, M$$

where O is the set of outcome vertices, X is the set of state vertices, A is the set of actuator vertices, D is the set of sensor vertices, C is the set of controller vertices, and M is the set of meta vertices. Likewise, the superset E is defined:

$$E \supseteq B, Y, Z$$

where B is the set of action edges, Y is the set of feedback edges, and Z is the set of conditional edges.

The structure of the Sim-graph shares striking similarities to a quiver- such as the potential to have multiple edges between two nodes, each with its own identity. However, the combination of disparate sets of vertices and edges, each representing a system component or behavior that may governed by incompatible mathematics, presents significant challenges to performing mathematical operations on the Sim-graph. Consequently, the Sim-graph is currently used to simply represent a visualization of the system and help formulate the behavior model in Simulink. The mathematical definition of the Sim-graph, however, does provide a starting point for future efforts intending to automate analysis of the system model.

4.1.3.2 The Simulink Behavior Model and Simulation

Due to the Sim-graph's potential limitations with respect to automated analysis techniques, it becomes necessary to use simulation for identifying appropriate resiliency strategies. Simulink provides the necessary tools for simulating the structure and behavior described in the Sim-graph without the difficulties associated with the mathematics of the graphical representation. Simulink's and Stateflow's combinatorial and sequential decision logic tools and other model elements enables the abstraction of some of the Sim-graph's complexity into a more easily executable form. More specifically, through a series of source blocks, mathematical operators, state machine diagrams, and flowcharts, this step of the methodology constructs a simulation of the system's intended behavior. The simulation of normal behavior is then used to determine where and how adverse behavior can be introduced, thus leading to the identification of appropriate resiliency strategies and their location within the system.

As previously mentioned, one of the difficulties associated with the definition of the S- graph is the diversity of what the vertices and edges represent. The Simulink model allows for these different types of vertices and edges to take on actual implementations of what they represent. For example, one particular controller in the system may follow a decision model that is describable in a truth table, whereas another controller operates based on a set of differential equations. Simulink enables both to be encoded to the desired level of granularity in the behavior model.

It should be noted that each behavior model and simulation is heavily dependent on the system in question and its associated mission. However, by using the Sim-graph as a starting point, each of the types of vertices and edges generally map to similar model elements within Simulink regardless of the system being modeled. Table 4 presents the types of Sim-graph elements mapped to a Simulink model element that should sufficiently describe its behavior for most applications.

By using Simulink source blocks and state machine diagrams to represent the state vertices, the simulation takes on a scenario-based format. This allows the SE team to generate conditions that produce the intended behavior of the system within the mission. Once the system's normal, or intended, behavior is represented by the simulation, then the SE team can explore ways to generate unintended or undesirable system behavior.

The SE team can take two approaches to producing undesirable behavior: by creating starting conditions based on the hazards defined in the consequence analysis, or by finding ways to generate the causal scenarios outlined in the consequence analysis. For each approach, the SE team documents the nature of any undesirable behavior that is generated, how it was generated, a list of potential mitigation strategies for that behavior, and the locations within the system for implementing those strategies. The potential mitigation strategies are based on the resilient design

Type	Corresponding Simulink Model Element
Outcome Vertex	Truth table
State Vertex	Source block and/or State machine diagram
Actuator Vertex	State machine diagram
Sensor Vertex	State machine diagram and/or math operator blocks
Controller Vertex	State machine diagram or truth table
“Meta” Vertex	State machine diagram
Type of Sim-graph Edge	
Action Edge	Embedded in truth table or state machine diagram
Feedback Edge	Inputs/outputs to and from sensor vertex model element
Conditional Edge	Inputs/outputs to outcome vertex truth tables

Table 4: A mapping of Sim-graph elements to a Simulink model element that should sufficiently represent its behavior in most applications.

patterns described by Jones and Horowitz for System-Aware cybersecurity [35].

The first approach defines a set of starting conditions that are either immediately hazardous, or likely to become hazardous. As an example, imagine that the model describes an autonomous vehicle. In this case, some of the state vertices would describe any obstacles in the vehicles path along with the vehicle’s current speed and heading. Following the scenario-based construction of the model, these variables would be programmable to be immediately hazardous at the start of the simulation, i.e. an obstacle directly in the path of the vehicle’s current heading and within the vehicle’s safe-maneuvering perimeter. In this scenario, the system’s behavior should account for this hazardous condition and attempt to mitigate the danger. If the system is unable to adequately handle such inputs, then the SE knows to investigate the introduction of some safeguards or resiliency measures to mitigate such situations. The selection of potential resiliency strategies and their location depends on the nature of the hazardous condition and varies from system to system.

The second approach to generate undesirable behavior in the simulation aims to generate hazards from within the model, rather than starting with hazardous conditions. More specifically, the initial starting conditions are such that the system should be expected to behave in its intended manner, however, the SE team changes parameters, noise levels, or other model elements with the intent of producing hazardous or unacceptable outcomes. Following the autonomous vehicle example described above, the intent would be to produce unsafe behavior from “normal” conditions. One potential method for doing so could involve introducing additional noise or bias into the system’s obstacle detection sensors. In a non-resilient system, this could easily result in the failure to detect and avoid an obstacle, thus creating a hazardous scenario and a potential unacceptable loss. Where these changes intended to produce unintended behavior occur within the system and what is being changed define the possible resilient design patterns that would be appropriate. For instance, in the above example, if increased noise in the obstacle detection sensors leads to undesirable behavior, then an appropriate resiliency strategy would be to include a noise monitoring algorithm and redundant backup sensors.

Both of these approaches should be used to describe all of the hazardous conditions and causal scenarios from the consequence analysis as appropriate- it is possible that not all the items from the consequence analysis are applicable to both approaches. Once all of the consequence analysis items are exhausted, the SE team should have a list of potential resilience strategies and the locations within the system for their implementation. At this point, the SE team should use their discretion to remove any strategies that address scenarios that might be unrealistic or are otherwise infeasible. Furthermore, it is possible that the list may contain some duplicate strategies; these items in the list should be merged and the number of duplicate entries recorded as this can be used as a measure of priority in strategy evaluation.

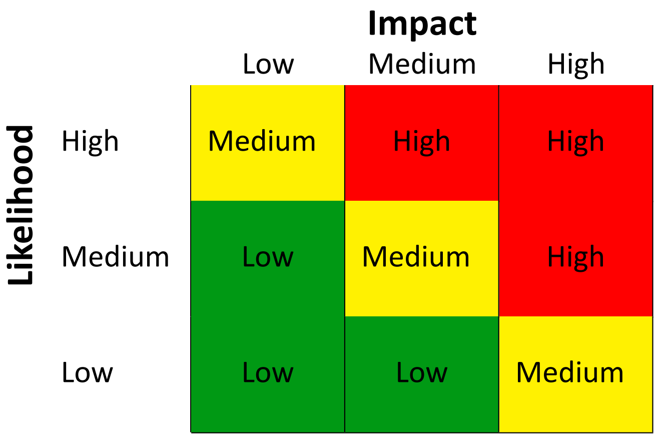
4.1.4 Evaluating Resiliency Solutions

The choice of which resiliency solutions to implement is a multi-criteria decision problem primarily involving the cost of the solution, the impact of the solution on the adverse outcome(s) to be mitigated, and the likelihood of the adverse outcome(s) occurring. How each of these factors, among the many others not mentioned, is dependent on the preferences and worldviews of the decision-makers. Furthermore, the cost of a resiliency solution, which includes the monetary value, the complexity of design, and the ease of integration into the system, varies greatly depending on the application. Thus, analysis of the cost factor is outside the scope of this thesis. However, the simulation and STAMP-based consequence analysis enable an evaluation of the adverse outcomes to be addressed by the solution as well as the solution's impact on those adverse outcomes. By taking advantage of the similarity of these two factors to the traditional definition of risk, the set of resiliency solutions can be prioritized into "risk" categories. These categorized solutions form a cost-agnostic recommendation of which resiliency measures to pursue. For every entry in the list of solutions identified based on analysis of the simulation, there is an associated list of adverse outcomes addressed by a particular solution. These adverse outcomes and the impact of those solutions form the basis of the "risk" measure based on the traditional definition [36]:

$$\text{risk} = \text{impact} \times \text{likelihood}$$

For the purposes of this application, impact is a measure of the number of adverse outcomes that a solution intends to address, the priority of those outcomes in the consequence analysis, and the effect of adding that solution on the operation of the system. This solution's effect on the system can be determined by adding in a representation of the solution to the simulation and comparing the results to the unaltered system if the nature of the solution allows. Otherwise, the effect must be judged qualitatively. Likelihood is a measure of the ease of achieving adverse outcomes in the simulation. More specifically, the number of changes to the simulation needed to achieve an adverse outcome and the severity of those changes. It should be noted that this definition of likelihood does not incorporate a probabilistic assessment of the ability of potential adversaries to create those changes in the system as such is out of the scope of this thesis. However, methods for creating such an assessment could easily augment the methods described here.

Given the nature of the factors that make up the impact and likelihood measures, quantitative metrics defining each dimension of the "risk" score are difficult, if not impossible to identify. Therefore, impact and likelihood are categorized into rankings of low, medium, and high. Thus,



Likelihood \ Impact	Low	Medium	High
High	Medium	High	High
Medium	Low	Medium	High
Low	Low	Low	Medium

Figure 7: The risk matrix prioritization framework for resiliency solutions.

the risk matrix framework can be readily applied to this application and resiliency solutions are categorized into low, medium, and high priorities for implementation [36].

After generating the set of recommended resiliency solutions in the risk matrix framework, all or a subset of the resiliency solutions can be applied to the system and the analysis iterated on the "new" updated system. A strength of the methodology presented in this section is the ability to refactor in resiliency solutions at multiple different steps. Solutions could be refactored into the initial system and mission descriptions or simply incorporated into the simulation model. Either approach offers greater confidence that all appropriate resiliency solutions are considered for a particular system.

Results of STRAT are shown in Section 5 for the Silverfish use case.

4.2 CYBOK and Security Analyst Dashboard

This section describes the development of two tools. The first, Cyber Body of Knowledge (CYBOK) is an information retrieval tool for systems engineers, security analysts, and requirements engineers. This tool discovers relevant attack information at the earliest possible stage of systems development using models of systems. The second, is called Security Analysts Dashboard, which is a combined user interface for CYBOK and graph transformations of SysML models. This tool presents the system topology, attack vector information, and the requirements diagrams defining the specification of the mission—therefore, allowing a common language between systems engineerings and security analysis within the MissionAware framework. Both these tools stem from past years research [37–39].

4.2.1 Role within the MissionAware Framework

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 (Figure 8). 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

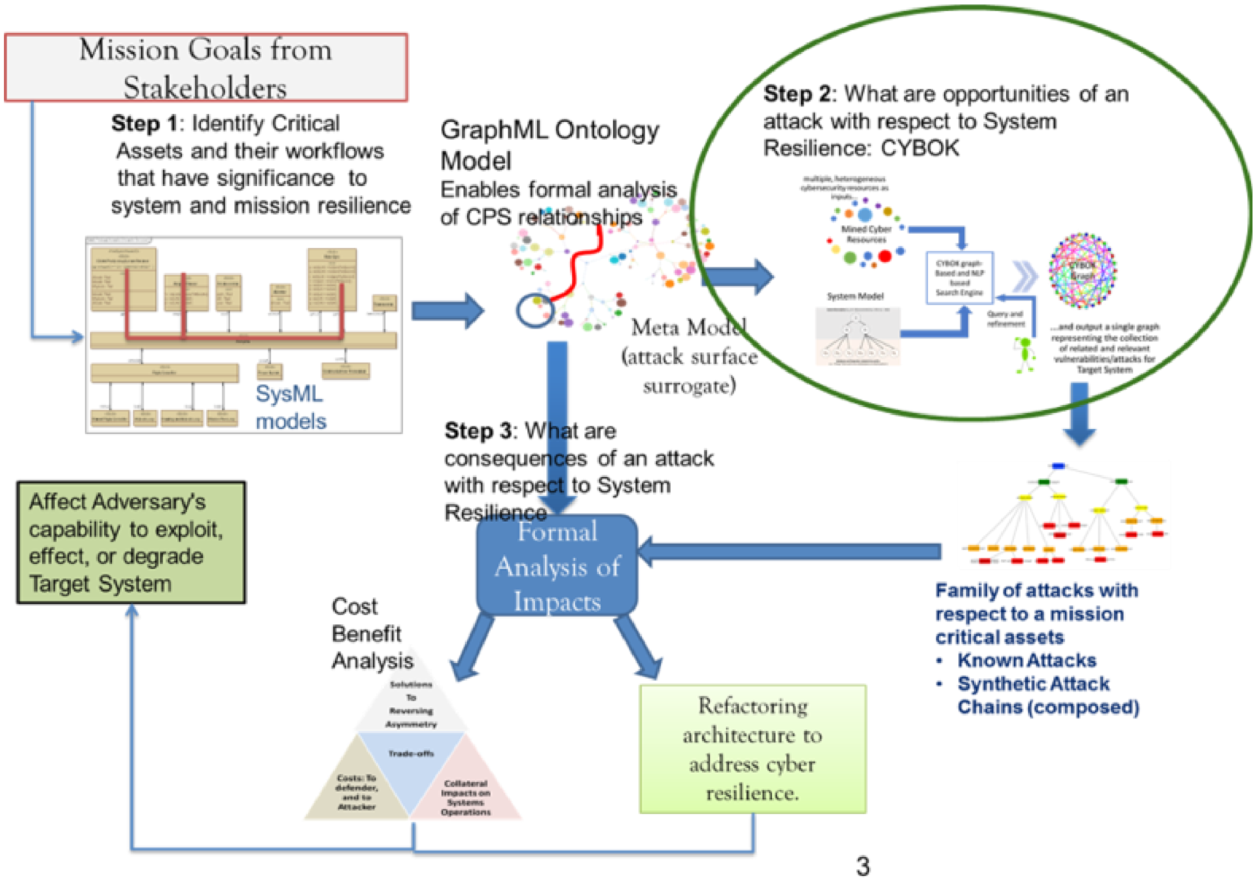

Mission Goals from Stakeholders
Step 1: Identify Critical Assets and their workflows that have significance to system and mission resilience
SysML models
GraphML Ontology Model
Enables formal analysis of CPS relationships
Meta Model (attack surface surrogate)
Step 2: What are opportunities of an attack with respect to System Resilience: CYBOK
Mined Cyber Resources
System Model
CYBOK Graph
Step 3: What are consequences of an attack with respect to System Resilience
Formal Analysis of Impacts
Affect Adversary's capability to exploit, effect, or degrade Target System
Cost Benefit Analysis
Solutions To Reversing Asymmetry
Trade-offs
Costs: To defender, and to Attacker
Collateral Impacts on Systems Operations
Refactoring architecture to address cyber resilience.
Family of attacks with respect to a mission critical assets
• Known Attacks
• Synthetic Attack Chains (composed)
3

Figure 8: Where Security Analyst Dashboard and CYBOK Fits into MissionAware

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 threats. The SysML modeling activity takes the output of the War Room and encodes mission critical information into workflow models to understand the potential threat space associated with the mission. This is the stage in which the tools; that is the dashboard and CYBOK, are used to help the system analysts and security engineers gauge the relevant vulnerabilities (and associated attacks) of the system and threat actors.

Through the Security 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 described in the architecture section (Section 4.2.2.2), 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 is consequential to a system and its mission.

4.2.2 CYBOK

To secure systems from emerging threats, systems engineers and security analysts alike need to integrate an attacker's view of vulnerabilities into their design, development, and analysis process—as early as possible. This is a basic tenant in MissionAware cyber security: a system's cyber security approach requires taking the attacker's perspective and relate these attacks to possible consequences to best understand how to strategically defend a system. To date, this attacker perspective activity has been a largely a manual process conducted by subject matter experts who examine a system and identify possible vulnerabilities. Moreover, these "red team assessments" tend to occur at later stages in the design and implementation lifecycle—where security modifications are more costly to implement and overall less effective. This does little to help the designer in trying to establish operational assurance early on in the system's development phase, and ultimately makes it more difficult for the security analyst later on.

Specifically, CYBOK is a tool that takes as input a graph system model, and uses it to identify known attack patterns, weaknesses, and vulnerabilities pertaining to the system by taking advantage of existing knowledge bases. CYBOK utilizes open databases, catalogs, and repositories used frequently in the threat sharing community. The aim of CYBOK is the creation of a tool which curates cyber security domain knowledge, for example, CAPEC, CWE, CVE, to provide usable information to both the security analysts and system engineers.

The general contributions of CYBOK are the following:

1. CYBOK is a multi-view search engine on how to relate threat information in a systems model context. It views the diverse set of security data 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 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.
3. CYBOK's information retrieval is *driven* by the system perspective—a SysML model, mission requirements, and operational assurance needs. Information from the SysML model of the system is distilled into a graph schema, encoded in the standard GraphML format. This is done automatically through a separate tool, graphml_export.
4. The results obtained by CYBOK can be easily examined, iteratively modified and decomposed and disseminated among the designers throughout the system lifecycle process.

4.2.2.1 Datasets

At present, we integrate three databases; that is, CAPEC, CWE, CVE, into the CYBOK search engine. Each of which serve different roles in cybersecurity analysis. Specifically, CAPEC, CWE, and CVE inform about attacks, weaknesses, and vulnerabilities, respectively. Therefore, each provides a different perspective on the security posture of a system. In addition to these diverse focal points

Table 5: Key features of each of the attack vector datasets.

Cybersecurity Resource	Focus	Representation	Size	Known Relationships	Data Format
CAPEC	Pseudo-ontology of Attack Patterns	Hierarchical Graph	527	Links to CWE & CVE	Human readable text, common technical words
CWE	Pseudo-ontology of Weaknesses and Vulnerabilities	Hierarchical Graph	806	Links to CAPEC & CVE	Human readable text, common technical words
CVE	Repository where vendors may announce vulnerabilities found in their software	Instance-based	113,098	Vendors using CPE may use the CPE Name for the affected software version(s); Links to CWE	Brief human readable descriptions, with additional info such as CVSS scores
CPE	Provide universal identifiers for software platform (single or multiple versions), as requested by Vendors	Instance-based	177,432	Used by CVE	Specially formatted; See the CPE specification for details; Uses platform-specific names
Exploit-DB	Code repository for PoC cyber-attacks	Organized by Target Platform	40,843	N/A	Program code; some human readable text

in the cybersecurity domain, instances of these datasets have direct relationships to one another and often relate to the same concepts from different viewpoints (Table 5). In addition to CAPEC, CWE, and CVE, we also list CPE and Exploit-DB.

CAPEC, is a pseudo-ontological hierarchy of attacks. It describes these 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 contained 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 that they target, and few refer to CVE 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. Like CAPEC, it is pseudo-ontological, providing a high level understanding of each of the concepts leading to vulnerability. It is tightly related to CVE, which describes specific platforms that have vulnerabilities. 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 is, instead, a database of specific platforms which provides a standard naming convention for those platforms to assist in vulnerability assessment, and which is used by CVE. At present this can be inferred from matches to CVE instances, but could be further integrated in the future, possibly giving an alternate route to modeling where CPEs are included in the model where available. CVE provides real instances of weaknesses from CWE becoming vulnerabilities, providing a platform-

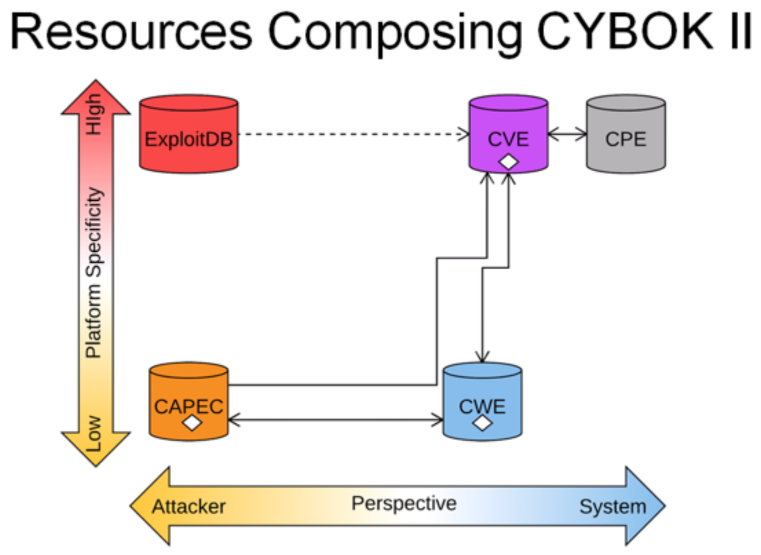


Figure 9: Illustration of the perspective each dataset has on the problem of cybersecurity.

specific perspective on how weaknesses occur. Even though Exploit-DB is not well connected to these other datasets, we include it because it is related to CAPEC in much the same way as CVE is to CWE—it provides real instances of attacks against specific platforms. Figure 9 illustrates the perspectives each of these databases has and how they are related.

Via the web, CAPEC, CWE, and, CVE can be searched on their respective websites using simple text-based queries, however this is inefficient for complex systems. In this context we need to be able to perform numerous searches on each component and interaction modeled in our system. Since no single one of these provides a complete picture on the system itself, the faults leading to vulnerability, and the attacks that can leverage such faults, it was deemed necessary to include each of these datasets in order to accomplish such a holistic perspective. We developed a search engine that incorporates all three of these datasets, and takes advantage of the interconnectedness they provide, in order to allow for efficiently identifying the threats to a target system.

4.2.2.2 Architecture and Implementation

CYBOK is implemented with a top-level search handler that takes incoming queries and search parameters and feeds them into two lower-level search processes, the index handler and the TaxaScore handler. The index handler performs a text-based search with the option of scoring instances with TF-IDF weighting, a well-established scoring method in the natural language processing literature. The TaxaScore handler processes parameters for how taxonomic scoring is to be performed and takes results from the text-based search and scores the ancestors and descendants of CAPEC and CWE instances to flesh out the families of threats associated with the matched instances.

This search process is wrapped in a command-line interface through which the Security Analyst Dashboard communicates with the CYBOK search engine. Through the command-line interface, inputs in the form of single queries, GraphML models, and parameter settings may be processed and searches may be performed, outputting relevant threat information to the provided input. The command-line interface also provides a mechanism through which attack surface and exploit chain analysis can be performed, with these results being output in GraphML and to command line, respectively. The architecture that accomplishes all the above functions is depicted in Figure 10.

The implementational aspects of the CYBOK architecture are as follows:

1. Command-line interface for executing CYBOK functions, which can be used directly or through the Security Analyst Dashboard.
2. Build and update process—a process that can be executed at command-line which downloads CAPEC, CWE, ans CVE from the web, processes the XML documents, and builds the necessary search index and taxonomies to be used by the CYBOK search process.
3. A text-based search engine with parameters for text-based weighting and what sources to search and/or report.
4. A graph-based search engine with numerous parameters controlling scoring of ancestors and descendants of matches to CAPEC and CWE instances, used to map a match of a threat to its more general and specific variations.
5. Search commands at command-line accepting either a single query or a GraphML model which can be used to run the text-based, and optionally graph-based, search processes, outputting results to CSV and GraphML.
6. Graph-related methods for processing GraphML models, attack surface analysis, and exploit chain analysis.

Together these features comprise a customizable search engine which can be used by the Security Analyst Dashboard and the security analyst to perform model-based threat assessment. In the following paragraphs we briefly describe the architecture of each of these components.

Command-line interface. CYBOK is operated via a command-line interface which manages various parameters for what the input will be and how the search will be done. It can be run on individual queries using the `-search` flag, or on a model using the `-input` flag. In either of these instances, their are additional flags for enabling and fine tuning the use of taxonomic scoring. There is also a flag, called `-target`, for finding possible exploit chains. Also, from the command line, the flag `-update` can be used to download CAPEC, CWE, and CVE form their sources, extract the data from these, and build the search index and taxonomies.

Search Handler. The search handler performs top-level logic of the CYBOK engine, handling the parameters of the underlying index and TaxaScore Handlers and controlling the flow of data between them when performing searches. The key value of this component is that it separates the logic defining text-based searches and that of taxonomic scoring, while also controlling the high-level search logic determining what sources are to be searched and reported, and whether results of text-based searches are to be passed into TaxaScore or not. It ensures that data output from CYBOK is consistent in either search case, with or without taxonomic scoring.

Whoosh/Index Handler. The index handler implements the core functionality underlying CYBOK's text-based searches. The text-based search process is implemented with the open-source

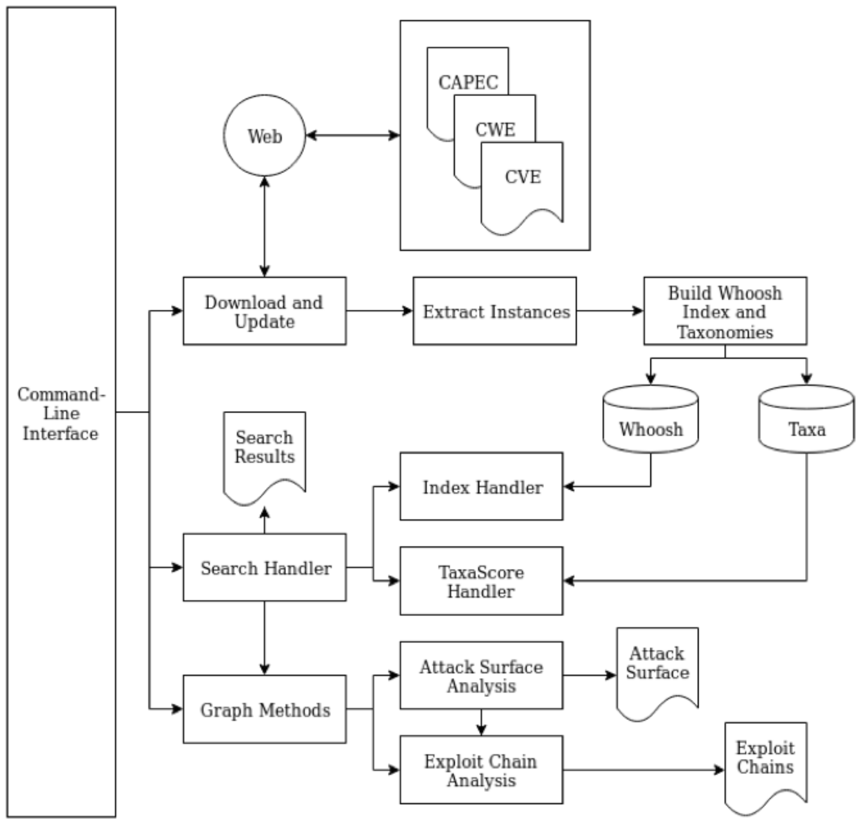


Figure 10: Architecture of CYBOK.

python library Whoosh, which handles indexing and searching of documents. The basic premise of a text-based search is that you find documents that share matching terms with the input query, in this case a system description, and you score them according to how well they match. The index handler manages the location of the Whoosh index to be searched, lookups this index, and generally records how text-based searches are to be performed. Namely, it allows for the user to select whether to use TF-IDF weighting or not, which is useful for determining the role taxonomic scoring has on the overall score produced by CYBOK. By controlling the text-based searches independently of taxonomic scoring, the index handler allows us to modify the text-based search module without interfering with downstream processes.

TaxaScore Handler. TaxaScore is a novel scoring process targeted at taxonomic datasets such as CAPEC and CWE. These datasets have the property that ancestors of an instance, its parent, parent's parent, and so on, describe more general forms of the threat concept, whereas their children represent more specific variations of the same concept. Accounting for this property, the key idea behind taxonomic scoring is that if we know with some certainty that an instance is relevant to our search, we can infer that so too are its generalizations and possible more specific forms. Thus, TaxaScore implements a scoring mechanism where the user can define the weight contribution that matches what their ancestors and their descendants should have when considering the value of a threat family with respect to a query or system description.

TaxaScore applies weights in three ways (Figure 11). First, a matched instance receives a score

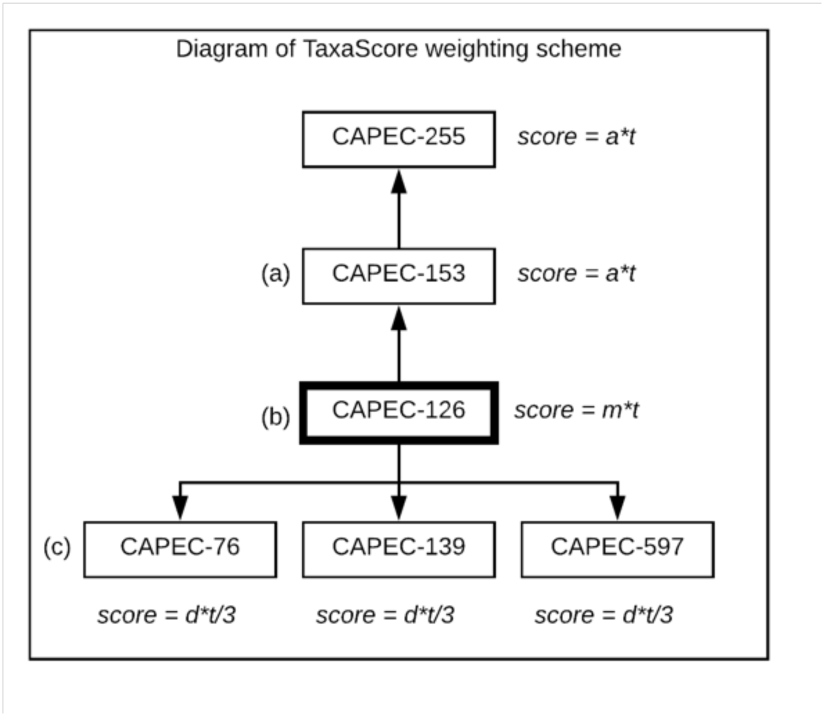


Figure 11: Diagram of TaxaScore scoring breakdown for a match with score *t*. (a) Shows the score an ancestor receives from the match, (b) shoes the score a matched instance receives, and (c) shows what score each of the children will receive.

equal to the weight of the matched, *t*, scaled by a constant *m*. Each ancestor of a match receives a score of the initial weight *t* scaled by a constant *a*. Lastly, a score of *t* times *d* is equally and recursively subdivided among children. After scoring all matched instances of a taxonomy, an instance's score is the sum of scores it received from ancestors, descendants, and from being matched directly.

In this way, TaxaScore uses the semantic understanding of the CAPEC and CWE graphs to extract out the broader context in which attack and weakness entries reside, informing the analyst of the full nature of matched threat families. As opposed to returning disparate instances of CAPEC and CWE, disconnected from the branches they are part of, TaxaScore ensures that each generalization of a matched threat concept is represented in the results of a search. The TaxaScore Handler is in charge of processing parameters for how taxonomic scoring is to be done and performing taxonomic scoring on CAPEC and CWE instances. In the current build, TaxaScore is available from command line in a separate branch or out code repository, but is not yet integrated into the Security Analyst Dashboard.

4.2.2.3 GRAPH METHODS

CYBOK contains a handful of methods for handling the GraphML file that stores the system model, and for assessing threats against that model. Two key parts of this are the methods involved in computing the attack surface of the model and the potential exploit chains throughout the system topology.

Attack Surface Analysis. The attack surface of a system is comprised of those pieces of hardware and software which can can serve as entry point for an attacker, or in other words, what is accessible from the outside either physically or virtually. By incorporating the "Entry Points" attribute into the model, the user can tell CYBOK which parts of the model they consider to be accessible, and CYBOK will identify which of these, if any, have possible attacks, weaknesses, and vulnerabilities to consider. By using the `-input` flag from command-line and inputting a model, CYBOK automatically will perform searches over all attributes, and will then construct the attack surface in GraphML from the model and those "Entry Points" attributes which returned results, outputting this GraphML file alongside the search results.

Exploit Chain Analysis. Exploit chains are possible paths through a system that can be attacked in sequence in order to go reach a target subsystem. CYBOK provides a tool that attempts to predict possible exploit chains by finding all possible paths through the system from its attack surface to a user-selected target node, where each node and edge on the path results in possible threats from a search in CYBOK of its attributes. When performing a search with the `-input` flag, exploit chains can additionally be provided by using the `-target` flag and providing the name of a component in the system. These two graph-based processes may give valuable insight to the analyst about how the system may be attacked, according to how the model has been defined and the threats which have been identified.

In summary, these components give CYBOK the capability to perform model-based threat assessment by matching threats from CAPEC, CWE, and CVE to user-provided descriptions contained in a model or in a single query. The numerous parameters available for fine-tuning TaxaScore allow the analyst to control how much weight to give to family members of matched threats, and the attack surface and exploit chain analyses allow for the prediction of possible paths that can be taken by an attacker to disrupt the system, thereby informing on possible mitigations that need to be made.

4.2.2.4 USAGE

CYBOK constitutes the core engine of the Security Analyst Dashboard, being in charge of identifying relevant threats to a system model. The search process done by CYBOK is a multiple step process that involves a text-based and a graph-based search. In Figure 12, we show the process and which paths are followed when doing a basic search (`-search`, in green), a model search (`-input`, in orange), and a model search with the exploit chain analysis (`-target`, in red). The `-input` and `-target` flags are additive with respect to the basic search and model search processes, respectively. This diagram shows how either a single query or multiple queries taken from a model are processed by CYBOK to report relevant threats, and optionally construct the attack surface and exploit chains of an input model.

In its most basic form (using the `-search` flag), a text-based search is done using the Whoosh library. Optionally, TF-IDF weighting can be used at this step to report the similarity of returned documents to the query by using the `-use_tfidf` flag. Following the text-based search, a tax-

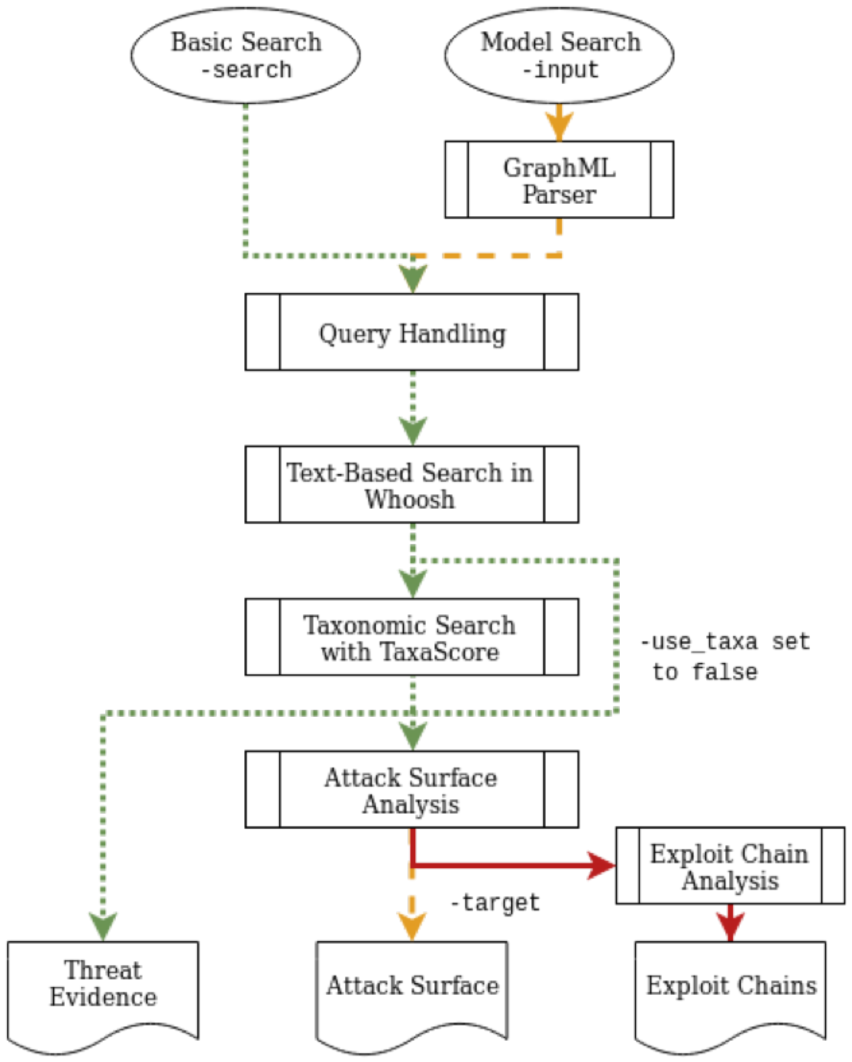


Figure 12: Process diagram of CYBOK. Green dotted path denotes basic search process (`-search` flag). Orange dashed path indicates the additional attack surface analysis that occurs when doing a search of a model (`-input` flag). Red path indicates the exploit chain analysis that occurs when the `-target` flag is included in a model search.

onomic search can be done by setting the `-use_taxa` flag to true. In this case, TaxaScore takes the related weaknesses to any matched CVEs, as well as matched CAPEC and CWE instances, and scores the ancestors and descendants of these instances to additionally report the more general and specific forms of matched threats. There are additional flags determining the weights applied to this scoring process for configuring TaxaScore. The results of this process can then be output in a CSV format where each row presents a matched instance, its score, relationships, and contents.

When performing a model search with the `-input` flag, CYBOK accepts a GraphML file storing the model as input. It iterates over nodes and edges, performing a search on each text attribute found. This behaves according to the same process as the basic search, but outputs a CSV where rows indicate which system component/edge and attribute produced each result. Additionally, the model search outputs a GraphML file with the attack surface, computed from “Entry Points” attributes which matched threats in the databases. If the `-target` flag is included with the name of a component in the model, the attack surface and the obtained search results will be used to determine all paths in the system from entry points to the target component for which threats were found on each node and edge.

4.2.3 Security Analysts Dashboard

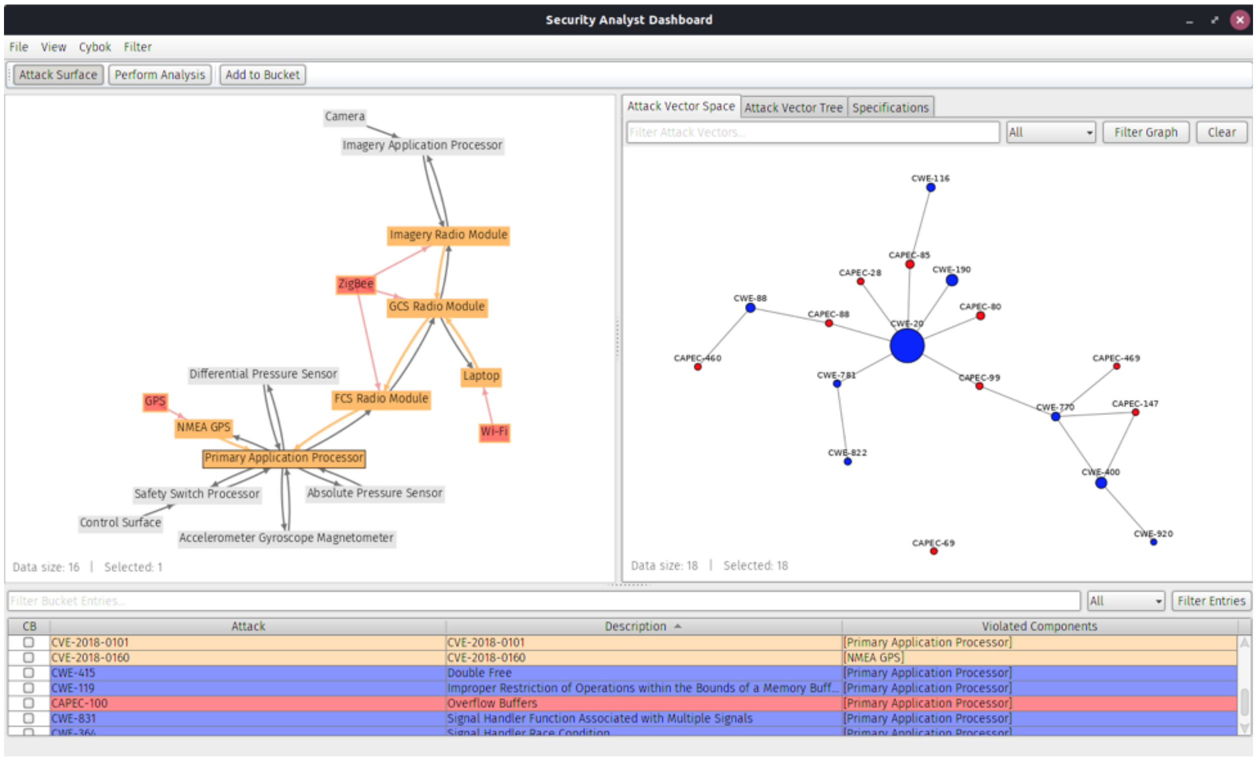
The SysML model produced by the War Room exercise describes the system in both a graphical, relationship-based manner, as well as each of its components and interactions in natural language. Since a goal of MissionAware is to determine the relevant threats to this system model, it presents a need for model-based threat assessment at this stage. To accomplish this, we have developed the Security Analyst Dashboard in order to mediate the relationship between the model and the threat assessment process. The dashboard consists of a robust UI with a number of tools for visualizing and editing the model (outside of SysML), as well as CYBOK, the search engine underlying the threat assessment process. With CYBOK to identify the threats associated to the model and the dashboard to assist in visualizing these results with respect to the model, the Security Analyst Dashboard is capable of informing the analyst about the threats facing the system and informs where mitigative actions might be necessary.

There are a number of important features implemented in the Security Analyst Dashboard which help it to perform model-based threat assessment and inform the analyst on the security posture of the system. These are:

1. **System Topology View.** A view which allows the analyst to examine and modify the system model, facilitating assessment of the model, and which can be used alongside threat assessment results to examine possible exploit chains through the system topology.
2. **System Specification View.** A hierarchical visualization of the mission requirements specification which can be used alongside threat assessment results to trace violation of mission requirements.
3. **Attack Vector Visualization.** A tool for visualizing and filtering results of the threat assessment in three distinct forms, (1) a graph view illustrating each threat entry and its relationships with other threats as a node and edges, (2) a list-tree view giving summary information about each entry and allowing to delve into its interconnected instances, and (3) a tabulated “bucket” view showing attributes and contents of each entry which allows the analyst to select important results of the threat assessment into a container which can be exported to CSV.

Together, these features give the Security Analyst Dashboard the mechanisms needed to allow an analyst to explore the system model and the threats associated with it, as determined through the two-part search process done by CYBOK (Figure 13).

The results produced by CYBOK via the basic and model-based search are used by the Security Analyst Dashboard to provide the analyst with the relevant threat information and possible attack paths that the system faces according to the descriptions put into the model.



CB	Attack	Description	Violated Components
☐	CVE-2018-0101	CVE-2018-0101	[Primary Application Processor]
☐	CVE-2018-0160	CVE-2018-0160	[NMEA GPS]
☐	CWE-415	Double Free	[Primary Application Processor]
☐	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buff...	[Primary Application Processor]
☐	CAPEC-100	Overflow Buffers	[Primary Application Processor]
☐	CWE-831	Signal Handler Function Associated with Multiple Signals	[Primary Application Processor]
☐	CWE-364	Signal Handler Race Condition	[Primary Application Processor]

Figure 13: A screenshot of the Securing Analyst Dashboard, showing a system topology with projected attack surfaces and exploit chains, the Attack Vector graph view, and the bucket. Each feature of which will be mentioned later.

4.2.3.1 System Topology Model View

The system topology model describes the design of the model under analysis (Figure 14). The model includes the individual components used, attributes that help describe their function, and edges that describe how they interact with the other components of the design. Specifically, the dashboard looks for the *Entry Point*, *Device*, *Operating System*, *Software*, and *Firmware* attributes.

Occasionally, the analyst may want to make minor modifications to their designs to quickly see the effects of changing certain components without having to constantly switch between programs. To address this issue, a simple model editor is included (Figure 15). This model editor allows the changing of an components name, attributes, and edges. Once modifications have been made, the analyst may redo the attack vector analysis facilitated by cybok. Once the analyst finds a design they are happy with, they have the option to export the model back into a graphml file for use elsewhere.

Additionally, the topology model view includes a feature to view attack surfaces and exploit chains. The attack surfaces show the entry points in which an attacker may exploit to affect the system. The attack chains show paths from entry points to a specific component showing through what paths an attacker may violate a component.

This view will enable the analyst to quickly determine the security state of the design and locate

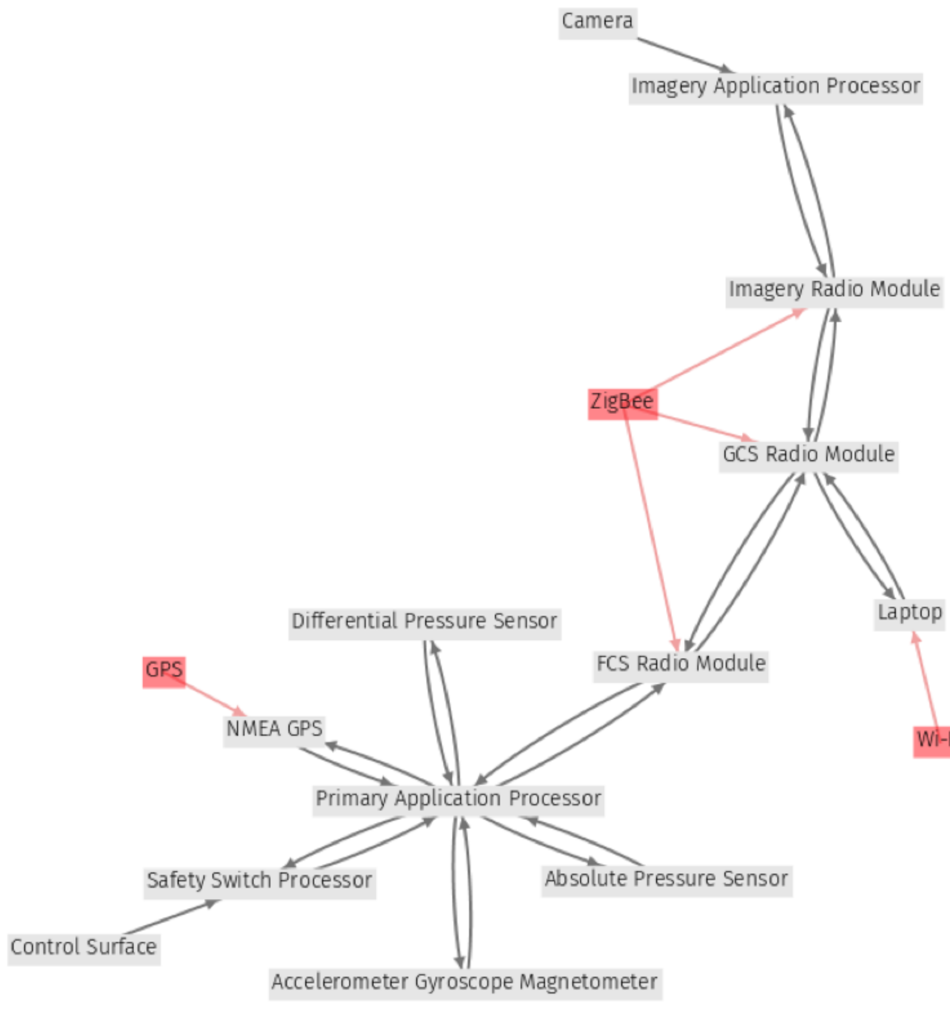


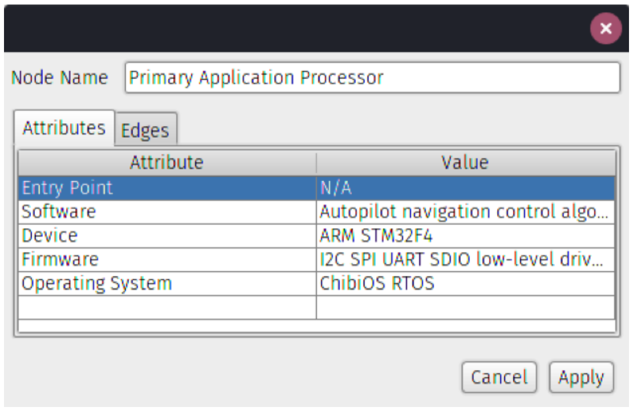
Figure 14: A screenshot of a system topology with projected attack surfaces displayed. The model was created in SysML and exported to GraphML using graphml_export, a plugin for MagicDraw.

areas where defenses or resilience techniques could be applied without requiring the investigation of every individual attack vector.

4.2.3.2 System Specification View

The System Specifications Model View provides a view from the MissionAware perspective, showing unacceptable losses, potential hazards during operation, and safety constraints. The requirements define overall operation, control actions, and necessary functionality. A custom hierarchical layout manager divides the provided specifications in three different groups:

1. *Mission level requirements* describe the overall operation of the design.


Node Name: Primary Application Processor

Attributes | Edges

Attribute	Value
Entry Point	N/A
Software	Autopilot navigation control algo...
Device	ARM STM32F4
Firmware	I2C SPI UART SDIO low-level driv...
Operating System	ChibiOS RTOS

Figure 15: Screenshot of the model editor for a specific component. Showing the component name and the associated attributes that describe it.

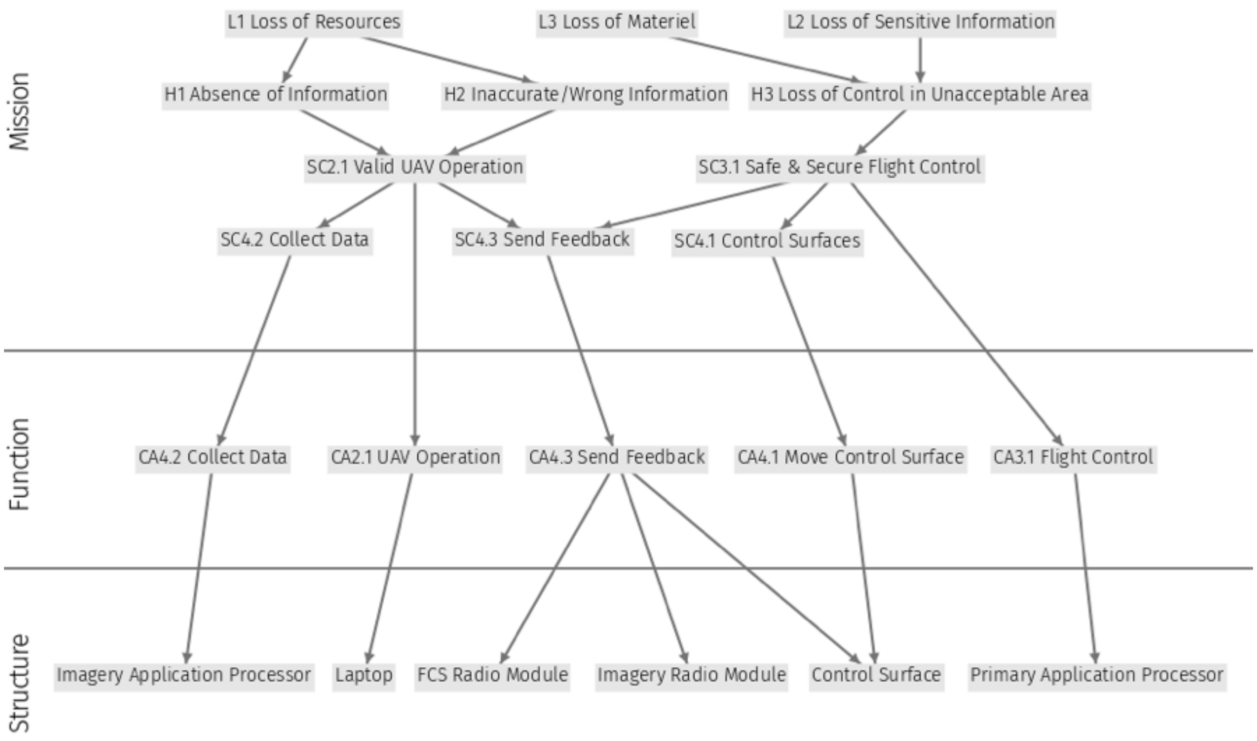
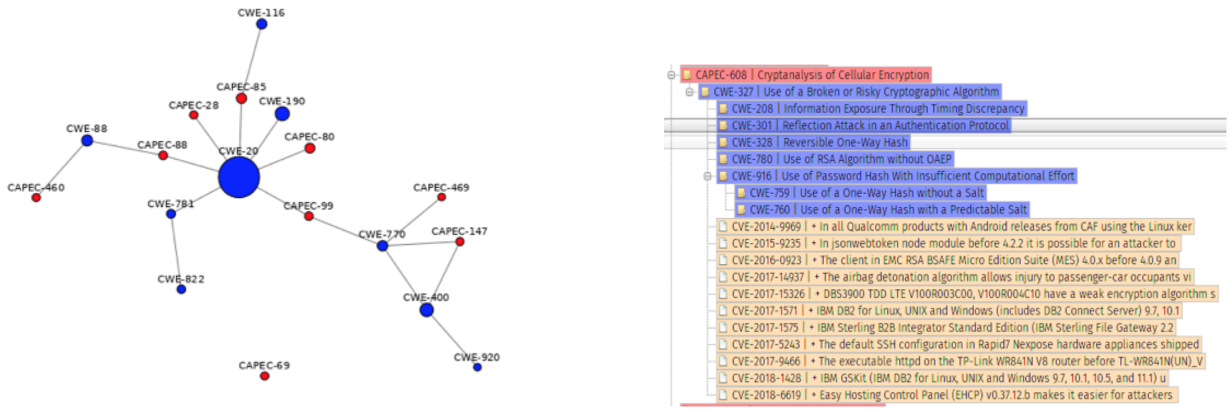

Mission
Function
Structure
L1 Loss of Resources
L3 Loss of Materiel
L2 Loss of Sensitive Information
H1 Absence of Information
H2 Inaccurate/Wrong Information
H3 Loss of Control in Unacceptable Area
SC2.1 Valid UAV Operation
SC3.1 Safe & Secure Flight Control
SC4.2 Collect Data
SC4.3 Send Feedback
SC4.1 Control Surfaces
CA4.2 Collect Data
CA2.1 UAV Operation
CA4.3 Send Feedback
CA4.1 Move Control Surface
CA3.1 Flight Control
Imagery Application Processor
Laptop
FCS Radio Module
Imagery Radio Module
Control Surface
Primary Application Processor

Figure 16: Screenshot of the system specifications view.

2. *Functional requirements* describe the functions the design needs to perform.
3. *Structural requirements* is comprised of elements used in the system topology, justifying their use by what function they serve.

The specifications view uses the attributes *type*, which can be one of the values Mission, Function, or Structure, and *text* which describes the requirement. The *type* attribute is used by the layout manager to determine what group to place it. The *text* attribute is used by the overlay renderer to display the description of the requirement when the analyst mouses over the node.



- CAPEC-608 | Cryptanalysis of Cellular Encryption
  - CWE-327 | Use of a Broken or Risky Cryptographic Algorithm
    - CWE-208 | Information Exposure Through Timing Discrepancy
    - CWE-301 | Reflection Attack in an Authentication Protocol
    - CWE-328 | Reversible One-Way Hash
    - CWE-780 | Use of RSA Algorithm without OAEP
    - CWE-916 | Use of Password Hash With Insufficient Computational Effort
      - CWE-759 | Use of a One-Way Hash without a Salt
      - CWE-760 | Use of a One-Way Hash with a Predictable Salt
    - CVE-2014-9969 | • In all Qualcomm products with Android releases from CAF using the Linux ker
    - CVE-2015-9235 | • In jsonwebtoken node module before 4.2.2 it is possible for an attacker to
    - CVE-2016-0923 | • The client in EMC RSA BSAFE Micro Edition Suite (MES) 4.0.x before 4.0.9 an
    - CVE-2017-14937 | • The airbag detonation algorithm allows injury to passenger-car occupants vi
    - CVE-2017-15326 | • DBS3900 TDD LTE V100R003C00, V100R004C10 have a weak encryption algorithm s
    - CVE-2017-1571 | • IBM DB2 for Linux, UNIX and Windows (includes DB2 Connect Server) 9.7, 10.1
    - CVE-2017-1575 | • IBM Sterling B2B Integrator Standard Edition (IBM Sterling File Gateway 2.2
    - CVE-2017-5243 | • The default SSH configuration in Rapid7 Nexpose hardware appliances shipped
    - CVE-2017-9466 | • The executable httpd on the TP-Link WR841N V8 router before TL-WR841N(UN)_V
    - CVE-2018-1428 | • IBM GSKit (IBM DB2 for Linux, UNIX and Windows 9.7, 10.1, 10.5, and 11.1) u
    - CVE-2018-6619 | • Easy Hosting Control Panel (EHCP) v0.37.12.b makes it easier for attackers

Figure 17: Screenshot of the two attack vector visualization methods, the left showing the graph view, and the right showing the tree view.

The traceability between levels help the analyst do a *what if* analysis on the effects of a system element violations from the perspective of the specifications of the system; that is, by seeing if a system element is violated what higher-level requirements could also be violated.

4.2.3.3 Attack Vector Visualization

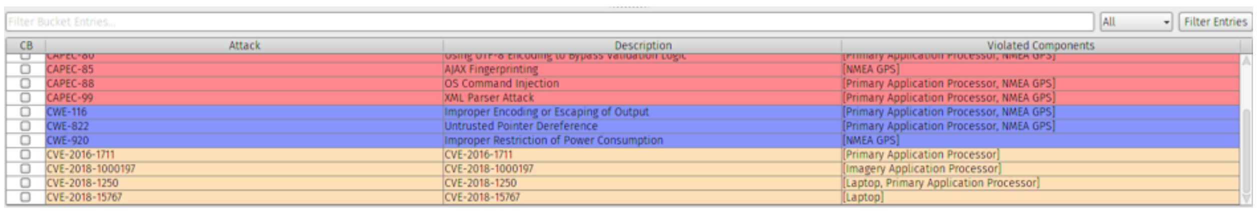
Attack vector visualization is an important tool that enables the analyst to easily navigate the expansive jungle of attack vectors that could potentially compromise a component of the system topology. To accomplish this, the analyst has a choice of two different visualization methods.

1. *Graph View*. A graphical representation of the attack vector space by showing both intra-related and inter-related connections between elements. The attack vectors are displayed where the CAPECs, CWEs, and CVEs are shown using red, blue, and yellow vertices respectively and the vertex size relates to the amount of connections associated. Analysts can interact with the view by moving each vertex around to help with visibility and by changing the perspective. By default, CVEs are hidden to help reduce the number of vertices shown without sacrificing important information. Shown on the left side of Figure 17.

2. *Tree View*. A structured tree representation of the attack vector space by showing the parent vertices as top level nodes that can be expanded to show the related children. This view uses the same color scheme as the graph view to maintain the consistency between views. Shown on the right side of Figure 17.

Both visualization methods include functionality that allows the analyst to select, delete, or open a web page with more information on a selected attack vector.

Another tool available to the analyst is the *bucket* (Figure 18). The bucket is a collection of attack vectors the analyst selected that they deem important to be further investigated or report to stakeholders. The collection is represented as a table where each row shows the attack id, description, and what components the attack vector potentially violates. Additionally, the contents



Filter Bucket Entries... | All | Filter Entries

CB	Attack	Description	Violated Components
☐	CAPEC-80	Using UTF-8 Encoding to Bypass Validation Logic	[Primary Application Processor, NMEA GPS]
☐	CAPEC-85	AJAX Fingerprinting	[NMEA GPS]
☐	CAPEC-88	OS Command Injection	[Primary Application Processor, NMEA GPS]
☐	CAPEC-99	XML Parser Attack	[Primary Application Processor, NMEA GPS]
☐	CWE-116	Improper Encoding or Escaping of Output	[Primary Application Processor, NMEA GPS]
☐	CWE-822	Untrusted Pointer Dereference	[Primary Application Processor, NMEA GPS]
☐	CWE-920	Improper Restriction of Power Consumption	[NMEA GPS]
☐	CVE-2016-1711	CVE-2016-1711	[Primary Application Processor]
☐	CVE-2018-1000197	CVE-2018-1000197	[Imagery Application Processor]
☐	CVE-2018-1250	CVE-2018-1250	[Laptop, Primary Application Processor]
☐	CVE-2018-15767	CVE-2018-15767	[Laptop]

Figure 18: Screenshot of the bucket.

of the bucket can be exported to a CSV file as a method of reporting the analysis to stakeholders or the contents of a CSV can be imported to populate the bucket. The CSV file includes the attack id, name, description, and the violated components. In the case of the graph and tree view, the filter bar also has the option of showing only the contents of the bucket.

Each of the visualization methods listed above also include filtering functionality. The attack vectors visible can be filtered based on the attack id, name, description, and by what components they violate. This allows the analyst to further narrow the visible attack vectors to what's relevant to the design.

4.2.4 Implementation

The current implementation of CYBOK is written in Python 3.6. XML documents are parsed using BeautifulSoup4. The text-based search index is implemented using the open-source library Whoosh. For handling GraphML files, NetworkX is used. The main version of the code also uses matplotlib and pygraphviz for visualization of the system topology and exploit chains.

The current implementation of the Security Analyst Dashboard is written using Java 8. The main user interface is created using Java's standard Swing library. The GraphML parsing is accomplished using the standard XML parsing libraries and rendered using then open-source library GraphStream. Interfacing with CYBOK is done by creating a python subprocess which allows operation much like how it would be called as if from a command prompt.

4.2.5 Summary

In summary, the Security Analyst Dashboard in conjunction with CYBOK provide tools in which both systems engineers and security analyst can assess a designs security state during the initial design process from the perspective of the attacker. The analyst can use the potential threats as identified by the attack vector threat assessment completed by CYBOK and displayed using the attack vector visualization tools alongside the unified system topology and specifications views, allowing the analyst to make informed defense and mitigation choices to protect the system.

5 Application to Silverfish

5.1 Description of System

CSRM [3] along with the STRAT tools is applied to a case study to test for efficacy of the new tools on a hypothetical new system. This system, known as Silverfish, is a theoretical weapon system deemed to be sufficient in terms of realistically representing a weapon system that could be used by the Army to perform a particular mission.

The system is defined as follows. The Silverfish system performs an area denial mission to aid the protection of a strategically sensitive location. More specifically, Silverfish deploys a set of 50 ground-based weapon systems, known as obstacles, that can engage unauthorized persons or ground vehicles within the denied area. The denied area measures up to approximately .16 square miles in size, with each obstacle capable of protecting a 300 foot by 300 foot area. A set of surveillance sensors including static infrared and video cameras and target characterization sensors, such as acoustic and seismic sensors, provide situational awareness by monitoring the area for persons and vehicles. An unmanned aerial vehicle also provides surveillance and early warning information by monitoring the periphery of the denied area. The Silverfish operator controls the obstacles and situational awareness sensors remotely from a nearby vehicle that can be maneuvered to give the operator "eyes-on" monitoring over portions of the denied area. The operator has control over the obstacles' armed or disarmed states and fire capabilities. He or she uses the situational awareness information available to determine target identity and the appropriate obstacle with which to engage the target. A wireless network relays the operator's commands from the control station to the obstacles. Furthermore, the operator has the ability to communicate with a command and control center to receive orders and additional situational awareness information. The system operates according to the following assumptions:

- Purpose: Deter and prevent, when and where necessary, via the use of rapidly deployable obstacles, adversarial tracked vehicles or individuals from trespassing into geographic areas that are close to strategically sensitive locations.
- Prohibited Area: 100 acres of open field space. At maximum speed a vehicle would take about 3 minutes to cross the prohibited area.
- Obstacle Deployment: About 50 obstacles are available to be distributed over the 100 acre protected area (each obstacle is designed to protect a 300x300 foot area). Each contains six (6) short-range sub-obstacles, each covering a 60-degree portion of a circular area to be protected.
- Operation: The operator, located in a vehicle that is operated close to the prohibited area (150 meters away), remotely controls individual obstacles and their sub- munitions, based upon sensor-based and operator visual surveillance of the prohibited area.
- Prohibited Area Surveillance: The operator is supported by obstacle-based acoustic and seismic sensors that can detect and distinguish between vehicles and people, redundant infrared sensors that can detect and track the movement of people and vehicles, and real-time Video/IR derived early warning information regarding people and vehicles approaching

the prohibited area provided by a UAV managed by the operator. The UAV is used to provide warning information.

- Obstacle design features: The obstacle-based sensors provide regular operator situation awareness reports when they detect a trespasser, reports on their location, their on-off status, and their remaining battery life. The obstacle confirms the acceptance of commands and the actual firing events.
- Infrared sensor configuration: A single pole-mounted IR sensor is assumed to be capable of providing surveillance of the entire protected area. A second sensor is provided for redundancy, and can be used to provide surveillance of areas that the single sensor is not able to observe.
- Requirements for Avoiding Errors: Concerns exist regarding activating sub-obstacles in cases where non-adversarial vehicles or people, by chance, enter the prohibited area. Concerns also exist about failing to fire munitions when an adversary is approaching a strategically sensitive location via the prohibited area. The operator, when possible, can use visual observations to increase confidence regarding fire control.
- Operator Functions: The operator can set the obstacles into either on or off modes and can cause individual or designated groups of obstacles/sub-munitions to detonate when in on mode. Obstacles can be commanded to self-destroy designated critical information in order to prevent adversaries from collecting such information for their own purposes. The operator also can launch a quad-copter drone (UAV) to provide video/IR based early warning information regarding potential trespassers of the protected area.
- Communications Systems: The communication system includes digital interfaces that support formatted data transfers between the operator's system, the UAV subsystem, the individual obstacles, the IR subsystem, and the C2 Center.
- Operator Control Station: The operator is provided with a vehicle-mounted computer(s) subsystem that provides situation awareness information including individual obstacle status, and sensor-based situation awareness information. The subsystem also provides computer-based entry and corresponding system feedback for control inputs from the operator.
- Command Center Controls: The C2 center digitally provides system control information for the operator (determines obstacle system on/off periods, provides warning of periods of higher likelihood of attack, provides forecasts of possible approach direction to the prohibited area, enables operation with/without UAV support, etc.).

A high-level, concept of operations representation of Silverfish is presented in Figure 19, using SysML. More details about the hardware and software design can be found in [3].

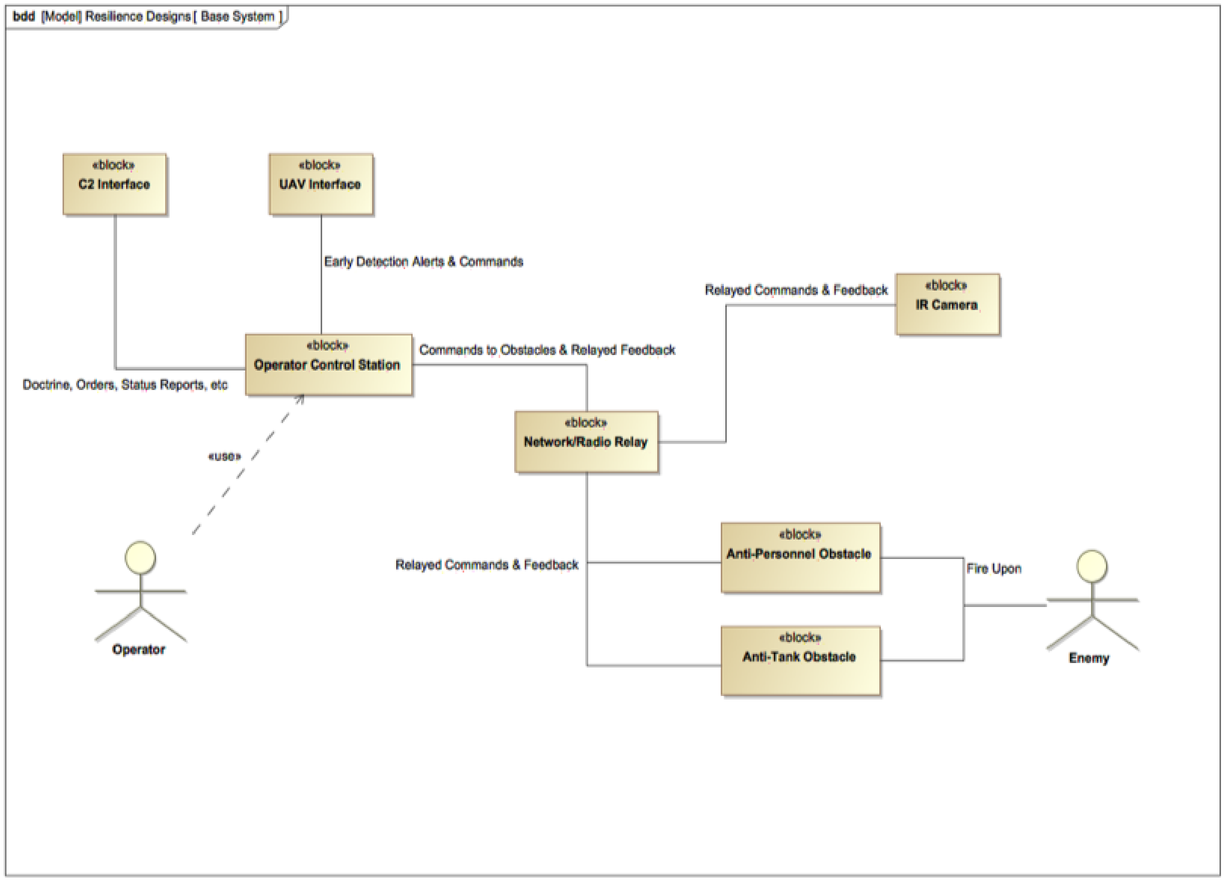

bdd [Model] Resilience Designs [Base System]
«block» C2 Interface
«block» UAV Interface
Early Detection Alerts & Commands
Relayed Commands & Feedback
«block» IR Camera
«block» Operator Control Station
Commands to Obstacles & Relayed Feedback
Doctrine, Orders, Status Reports, etc
«block» Network/Radio Relay
«use»
«block» Anti-Personnel Obstacle
Relayed Commands & Feedback
Fire Upon
Operator
«block» Anti-Tank Obstacle
Enemy

Figure 19: A Concept of Operations representation of Silverfish in SysML.

5.2 Application of STRAT

This section details an application of the approach described in Section 4.1 on the hypothetical US Army weapon system analyzed for the CSRM, known as Silverfish. Results are then compared with the recommendations of the CSRM to assess the compatibility of the methodology with existing techniques. The methodology presented in this section uses the same mission and system descriptions and Blue Team-defined unacceptable consequences as the CSRM to allow comparison of recommendations. As stated in Section 4.1, the tools used in this report do not necessarily require that this information be collected in the same way as the CSRM; however, it should be acknowledged that engaging the system owners increases the veracity of the collected information.

5.2.1 Mission and System Specification

The Silverfish system was initially developed to be a testbed for the application of the CSRM. Although it is a hypothetical system, the US Army Armament Research, Development, and Engineering Center (ARDEC) determined that the system is both representative of a system that could potentially be used by the Army and is suitable for the demonstration of cybersecurity techniques.

The assembled Blue Team, composed of members of the ARDEC, and the SE team developed the initial Silverfish mission and system descriptions through a series of iterations before agreeing upon the final description below [3].

The Silverfish system performs an area denial mission to aid the protection of a strategically sensitive location. More specifically, Silverfish deploys a set of 50 ground-based weapon systems, known as obstacles, that can engage unauthorized persons or vehicles within the denied area. The denied area measures up to approximately .16 square miles in size, with each obstacle capable of protecting a 300 foot by 300 foot area. A set of surveillance sensors including static infrared and video cameras and target characterization sensors, such as acoustic and seismic sensors, provide situational awareness by monitoring the area for persons and vehicles. An unmanned aerial vehicle also provides surveillance and early warning information by monitoring the periphery of the denied area. The Silverfish operator controls the obstacles and situational awareness sensors remotely from a nearby vehicle that can be maneuvered to give the operator "eyes-on" monitoring of the portions of the denied area.

The operator has control over the obstacles' armed or disarmed states and fire capability. He or she uses the situational awareness information available to determine target identity and the appropriate obstacle with which to engage the target. A wireless network relays the operator's commands from the control station to the obstacles. Furthermore, the operator has the ability to communicate with a command and control center to receive orders and additional situational awareness information. For the purposes of this thesis, the analysis and recommendations are limited to the components that are "owned" by the Silverfish system. This means that the command and control center, the UAV, and the vehicle have their capabilities and inputs to the system considered when identifying resiliency strategies, but changes to these systems are out of scope for analysis.

Following the framework described in Section 4.1, the criteria for mission success are simple: all unauthorized persons or vehicles in the denied area are engaged correctly for the duration of the mission. Mission failures result from unauthorized persons or vehicles successfully traversing the denied area or friendly fire incidents.

Following the finalization of the mission and system descriptions, the Blue Team and SE team met to develop a prioritized list of unacceptable consequences with respect to the Silverfish mission. The CSRM supplemented this meeting with SysML representations of the agreed upon mission and system descriptions. Each entry in the list of consequences received a priority based on the following Likert Scale:

1. Unacceptable and highest priority to provide resiliency
2. Avoid as long as resiliency solution does not over-complicate operation
3. Would like to avoid, but solution needs to be incremental
4. Lowest priority, low-cost, simplistic solutions should be considered.

Within each prioritization level, the consequences were further ranked based on the Blue Team's perception of their severity. For each consequence, the potential targets of an attack that would produce that outcome was identified, along with the potential method for completing the attack.

Finally, the types of inappropriate control actions that would be associated with that consequence were identified using the following scale:

1. Providing a control action causes a hazard
2. Not providing a control action causes a hazard
3. Incorrect timing or improper order of control actions causes a hazard
4. A control action is applied too long or stopped too soon.

The output of this meeting is presented below in Table 6.

Likert Rank	Consequence	Attack Target(s)	Attack Method	Control Action Type
1.1	Inappropriate firings via manipulating operator commands	Operator control display, radio comm links	External, supply chain, insider	1, 2, 3
1.2	Delays in fire time (sufficient delay to cross field)	Obstacles, control station, radio comm links	External, supply chain, insider	2, 3
1.3	Delays in deployment	Obstacles, deployment support equipment	Supply chain, insider	2, 3
1.4	Deactivation of a set of obstacles	Obstacles	External, insider	1, 3
2.1	Delays in situational awareness	Operator display, sensors	External, insider, supply chain	1, 2, 3
2.2	Prevent or corrupt transmission of situational awareness data	Radio comm links, operator display, sensors	External, insider, supply chain	1, 2, 3
2.3	Gain information to help adversary navigate through field	Obstacle, operator control station	External, insider	2, 3
3.1	Reduced operational lifespan	Obstacle	External, supply chain, insider	1, 2, 3,
3.2	Prevent transmission/execution of non-firing commands	Operator display, obstacles	External, insider, supply chain	1, 2
4.1	Delays in sending/receiving C2 information	Operator display, radio comm links	External, supply chain	1, 2, 3
4.1	Delays in un-deployment	Obstacles	External, insider, supply chain	1, 2, 3

Table 6: The list of Blue Team-derived undesirable consequences.

The list of unacceptable consequences, along with the Silverfish mission and system description, form the ground truth from which all further analysis is based. Following the completion of this step, the STRAT does not require further involvement of the system owners (Blue Team) or other non-SE team members, unlike the CSRM.

5.2.2 Systems-Theoretic Consequence Analysis

As described in Section 4.1, following the definition of the mission and system descriptions and the identification of undesirable consequences, STRAT uses STAMP and STPA-Sec concepts to document unacceptable losses, hazards, and the system's control structure.

The mission and system descriptions defined the basic conditions for mission success and failure. Silverfish achieves mission success if no unauthorized agents traverse the denied area for the duration of the mission; mission failure occurs when unauthorized agents successfully traverse the denied area or obstacles are fired upon friendly forces. These two definitions translate into the following unacceptable losses or outcomes for this mission:

- L1 – Enemy forces or other unauthorized persons/vehicles traverse the denied area without the operator's knowledge or intent,
- L2 – Friendly forces, civilians, or other non-combatants are killed or harmed by Silverfish,
- L3 – Silverfish obstacles are fired without a valid target.

Unacceptable losses L1 and L2 clearly map to the stated mission of Silverfish; however, L3 was derived as an additional, lower priority unacceptable loss because of the implications it has on the outcome of the mission. As seen in the outcomes described in Table 6, the Blue Team is concerned about losing control of Silverfish or Silverfish not being able to operate as intended for the mission's duration- L3 describes a third end-result of such consequences that does not involve friendly fire or the immediate traversal of unauthorized agents through the denied area.

Following the definition of the unacceptable losses, the STRAT defines the hazardous conditions that could lead to an unacceptable loss. These hazards define conditions that do not immediately result in an unacceptable loss, but will lead to an unacceptable loss given an improper implementation of a control action or the presence of a worst-case environment. Table 7 defines a set of hazardous conditions, the worst case environment for those occur in, and the unacceptable losses associated with that hazard.

Next, the basic control structure of the Silverfish system is defined. Using the control loop format described in Section 4.1 and Figure 5, Silverfish is decomposed into its main controllers, sensors, actuators, and controlled processes. Based on the system description, Silverfish consists of an operator who controls the obstacles and visual sensors through a control station over a wireless network. This involves the operator overseeing three controlled processes: fire control, surveillance, and target characterization. The operator manages all three processes through the control station. The obstacles actuate fire control commands, the visual sensors actuate surveillance and target characterization, and the characterization sensors (acoustic and seismic) also enable target characterization. The sensors provide feedback to the operator on the three controlled processes via the control station. This basic structure is presented in Figure 20. It should be noted that the simplicity of this particular system is not necessarily shared by other systems.

Hazard	Worst Case Environment	Associated Losses
H1- Failure to fire correct obstacle	Imminent threat entering denied area	L1
H2- Incorrect obstacle armed or fired	Friendly in denied area	L1, L2, L3
H3- Wireless link to obstacles down	Imminent threat in denied area	L1
H4- Situational Awareness data inaccurate, delayed, or unavailable	Imminent threat entering denied area; friendly agent in denied area	L1, L2

Table 7: Hazardous Conditions that could lead to an unacceptable loss.

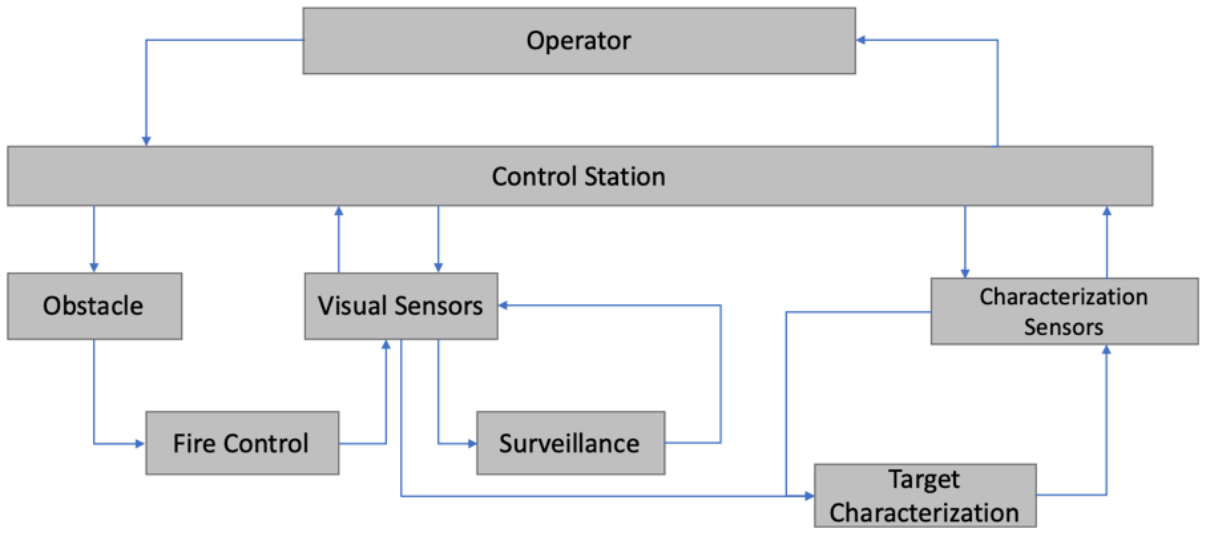


Figure 20: The basic control structure of Silverfish.

From this control structure, the control actions available at each hierarchical level are enumerated, and the conditions under which each control action contributes to a hazard identified. In the representation of the system described in Figure 20, the control actions available to operator are effectively identical to those available to the corresponding lower levels of the system. Consequently, those control actions for the other hierarchical levels of the system are omitted from the control actions in Table 8 as they would be redundant. Again, this characteristic is a result of the simple nature of the Silverfish system, and not indicative of other applications. Understandably, however, if the lower level controllers do not enact the operator's control actions accurately, then hazards are likely to occur.

The final step of the consequence analysis involves the generation of causal scenarios that describe the implementation of improper control actions. The undesirable outcomes defined by the Blue Team motivate the definition of causal scenarios associated with each control action. These causal scenarios help motivate the choice of appropriate resiliency measures in the next step by providing

Control Action	Not Providing causes hazard	Providing Causes hazard	Incorrect Timing or Order	Stopped too soon or applied too long
Operator Control Actions				
CA 1.1-Arm/Disarm Obstacle	Target in denied area- H1, H3	Friendly in denied area- H2	Target not in range- H1	Target not in range- H1
CA 1.2- Fire Obstacle	Target in range-H1, H3	Friendly in range- H2	Target not in range- H1	Target not in range- H1
CA 1.3- Adjust visual sensor field of view	Target not identified- H1, H3, H4	Target goes unidentified-H1, H4	Target unidentified-H1, H4	Target goes unidentified-H1, H4
Obstacle/Sensor Control Actions				
CA 2.1- Send feedback	Operator doesn't receive data- H1, H3, H4	Data is corrupted-H1, H2, H4	N/a	Target goes unengaged- H1

Table 8: Control actions and the conditions under which they would contribute to a hazard.

details on what might cause a control action to be implemented improperly. Table 9 presents control actions mapped with an associated causal scenario and the priority rank of the related undesirable outcome(s) from the Blue Team.

Control Action	Causal Scenario	Blue Team Outcome
CA 1.1	Legitimate operator control action overridden or altered due to cyber-attack on control station or network	1.1, 1.2, 1.4,
CA 1.2	Legitimate operator control action given, but improper due to misclassification of target	2.1, 2.2, 3.2, 4.1
CA 1.3	Legitimate control action overridden or altered due to cyber-attack	2.1, 2.2, 2.3, 3.2
CA 2.1	Cyber-attack causes delay, denial, or increased rate of control action application	2.1, 3.1, 3.2

Table 9: Causal scenarios for implementing an improper control action mapped to undesirable Blue Team outcomes.

5.2.3 Model-based Resilience Solution Identification

Model Construction

The next step in the STRAT begins with the development of the graphical system model. This model shares the same basic shape as the hierarchical control structure identified in the consequence analysis, but incorporates additional STAMP-related information. Using the definitions outlined in Section 4.1, the Sim-graph for the Silverfish system is presented in Figure 21. Each vertex and edge is color-coded to the types described previously.

The vertices labelled 1, 2, 3, 5, and 7 map directly to their corresponding blocks in the control structure shown in Figure 20. The vertices labelled 4 and 6 represent the physical states that define the presence or absence of an unacceptable loss. The obstacle state describes whether or not the obstacles are armed and whether or not an obstacle has been fired. Likewise, the denied area state describes any agents within the denied area and their location. The vertex labelled 8 represents the outcome matrix, which describes the presence or absence of an unacceptable loss. The edges labelled a, b, c, h, and i are the action edges, which describe the control actions or dynamics through which the parent vertex influences the target vertex. The edges labelled d, e, f, g, k, and l represent feedback from the parent vertex to the target vertex. Finally, the edges m and n represent the conditional edges that are the inputs to the outcome matrix.

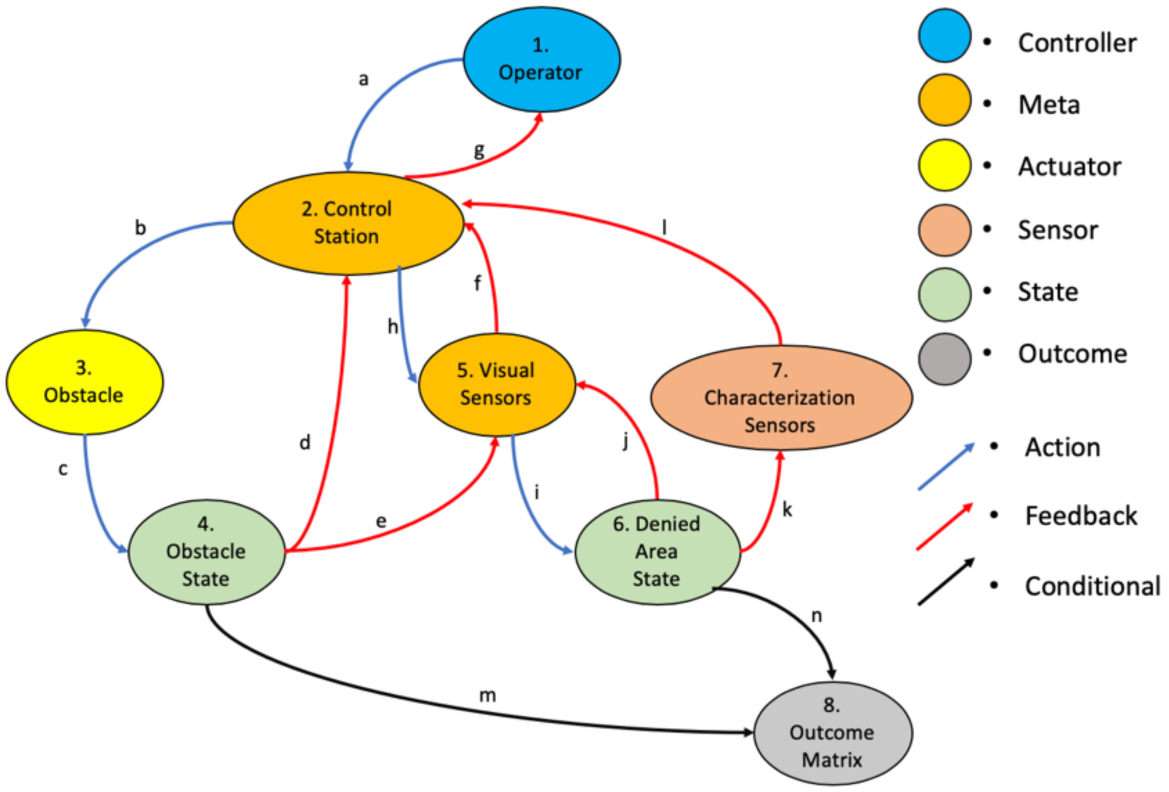


Figure 21: The Sim-graph for the Silverfish System.

As stated previously, at this point of development, the Sim-graph mainly serves as the foundation for the Simulink behavior model. Future research on the mathematics of the Sim-graph formulation could allow for further analysis on the system's control structure.

The Simulink model follows the construction guidelines defined in Table 4. For the purposes of this particular system, however, the operator and control station are represented as a single entity. This is because the control station and the operator follow the same decision logic, thus making separate model representations superfluous. A screenshot of the Simulink model is shown in Figure 22.

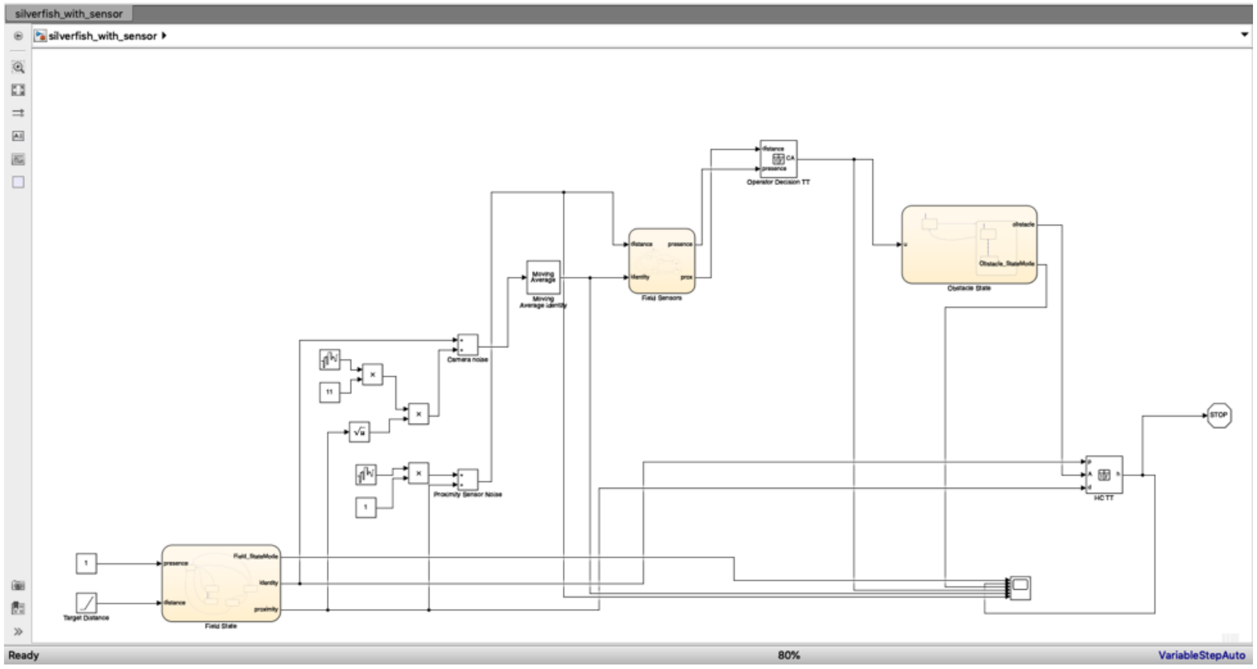


Figure 22: A screenshot of the Simulink behavior model.

The Simulink model follows the scenario-based formulation described in Section 4.1. That is, the state variables not controlled by the system, agent identity and proximity to an obstacle are defined as source inputs to the simulation. For the purposes of this application, agent identity is defined as a constant and proximity decreases linearly from an initial with time. These variables are the inputs to a state machine diagram that defines the "true" state of the denied area based on the values of the source variables. This "true" state of the denied area is defined as one of the following states in the state machine diagram (combinations of these states are not considered as the operation of Silverfish in such situations becomes dependent on the operator's specific rules of engagement):

1. No agent present in the denied area
2. A non-enemy agent in the denied area, but out of range of an obstacle
3. An enemy agent in the denied area, but out of range of an obstacle
4. A non-enemy agent in the denied area, and in of range of an obstacle
5. An enemy agent in the denied area, and in of range of an obstacle.

The state of the denied area is monitored by the surveillance and target characterization sensors, which introduces noise into the estimate of the state of the denied area. This estimation is represented by another state machine diagram with the same states defined above, however, the inputs are combined with Simulink noise blocks. The estimate of the state of the denied area forms the input to the operator's decision model regarding which control actions to take. This decision model is encoded in a truth table that maps the estimated denied area states to an appropriate control action. This truth table is presented below in Table 10.

Condition	D1	D2	D3
Agent Present in Denied Area	T	T	-
Confirmed Enemy Agent	T	T	-
Agent in Range of Obstacle	T	F	-
Control Action	Fire	Arm	Disarm

Table 10: A truth table representation of operator decision logic.

The resulting control action of the truth table then forms the input to the state machine diagram that represents the state of the obstacle. Each obstacle can be armed, disarmed, or fired. The state of the obstacle is then combined with the "true" state of the denied area to form the inputs to the outcome matrix. The outcome matrix is also defined as a truth table, mapping in Sim-graph states to consequence analysis losses and hazards, seen below in Table 11.

Condition	D1	D2	D3	D4	D5	D6	D7	D8	D9
Agent Present in Denied Area	T	T	T	T	T	T	F	F	-
Confirmed Enemy Agent	T	T	T	T	F	F	F	F	-
Agent in Range of Obstacle	T	T	T	F	T	T	F	F	-
Obstacle Armed	T	T	F	F	T	T	T	T	-
Obstacle Fired	T	F	F	F	T	F	T	F	-
Outcome	n/a	L1	H1	H1	L2	H2	L3	H2	n/a

Table 11: A truth table description of the outcome matrix.

Running the simulation in its baseline configuration always results in the "n/a" outcome defined in Table 11, which indicates that the system is operating as intended given a particular set of starting conditions.

Identifying Resiliency Solutions from Simulation Changes

As described in Section 4.1, changes to the simulation intended to create the hazardous conditions and unacceptable losses from the consequence analysis identify the locations for potential resilience solutions. For this particular system, the first approach to producing adverse outcomes-introducing hazardous starting conditions for the simulation does not apply. Since the sources variables describe the agent identity and proximity, the only possible hazardous starting condition would be an enemy agent in range of an obstacle. As stated in the previous section, the system's decision logic would immediately resolve the situation.

The second approach to producing adverse outcomes in the simulation, however, yields meaningful results. The causal scenarios identified in the consequence analysis and defined in Table V describe potential ways that improper control actions can lead to adverse outcomes. These causal scenarios can be categorized into three types of root causes:

1. Legitimate control actions made based on erroneous decisions
2. Legitimate control actions overridden by invalid control actions
3. Control actions blocked or delayed in implementation.

Using these three types of root causes as a basis for design, changes are made to the simulation with the intent of producing the hazardous conditions defined by the consequence analysis.

For example, adding bias to the estimate of an enemy agent's proximity to a particular obstacle can result in the failure to fire the correct obstacle- H1. Depending on the geometry of the denied area, as little as a 10% bias to the proximity estimate can result in the firing of an incorrect obstacle (assuming that the obstacles have a 50 meter range and the range of adjacent obstacles overlap by 5 meters). Following this example, each hazardous condition from the consequence analysis is mapped to a list of changes in the simulation that produce the outcome and their corresponding locations in the system in Table 12.

Outcome	Changes to Simulation to Produce Outcome	Location
H1	Negative bias or increased noise in identity estimate	Visual sensors, classification algorithms (if applicable)
	10% bias in proximity estimate (under right conditions)	Characterization sensors, control station log of obstacle locations
	Confusion of control actions between operator input and obstacle implementation	Control station, obstacle
	Incorrect reporting of obstacle state	Control station, obstacle
	No control input to obstacles	Control station, network
H2	Positive bias or increased noise in identity estimate	Visual sensors, classification algorithms (if applicable)
	Confusion of control actions between operator input and obstacle implementation	Control station, obstacle
H3	No control input to obstacles	Control station network
H4	Increased noise or bias added to identity and proximity estimates	Sensors, classification algorithms (if applicable)

Table 12: A mapping of simulation changes to the hazardous conditions they contribute to.

The combination of the type of change made to the simulation and the corresponding location within the Silverfish system drive the selection of resiliency solutions that would mitigate the hazardous condition mapped to the change. For example, a resiliency strategy that would address the lack of control input to the obstacles associated with the network would be the inclusion of a backup communication system to control the obstacles. Whereas a strategy that addresses the same change to the simulation, but a different location within the system, would be the inclusion of diverse hardware components within the control station that rotate responsibility for sending commands to the obstacle over the network. Following this pattern, resiliency solutions are identified for each of the remaining simulation changes and corresponding locations in Table 13.

Change to Simulation	Location	Resiliency Solution
Bias or increased noise in identity estimate	Visual sensors	Redundant camera system with lesser performance
	Classification algorithms (if applicable)	System parameter assurance
10% bias in proximity estimate (under right conditions)	Characterization sensors	Triple redundant acoustic sensors for increased confidence in proximity measurement
	Control station based log of obstacle locations	System parameter assurance
Confusion of control actions between operator input and obstacle implementation	Control station	Diversely redundant, hopping command sending capability
	Obstacle	Two-factor command authorization
Incorrect reporting of obstacle state	Control station	System parameter assurance
	Obstacle	Operational consistency checking for obstacle feedback
No control input to obstacles	Control station	Diversely redundant, hopping command sending capability
	Network	Backup communication network

Table 13: Potential Resilience Solutions mapped to simulation changes.

The list of potential resiliency solutions is now consolidated into a mapping of each solution to the locations for implementation, the hazardous condition(s) to be mitigated, and the associated Blue Team adverse outcomes. This mapping is shown in Table 14.

5.2.4 Evaluation of Identified Resiliency Solutions

Given the set of resiliency solutions identified in the previous step, each solution is now evaluated in terms of the risk-based framework described in Section 4.1. The impact of each solution is a factor based on the number of adverse outcomes addressed, the priority of those adverse outcomes, and the solution's effect on system operation. The priority of outcomes is defined as the average likert priority ranking of the associated Blue Team adverse outcomes. As stated in Section 4.1, some resiliency solutions may be possible to represent in the simulation. For this particular application, the only solution solutions immediately representable within the simulation are the ones addressing increased noise or bias within the system's sensors. For example, the triple-redundant acoustic sensors lowers the amount of noise perceived by the system, and allows for a

Resiliency Solution	Location	Mitigated Hazard(s)	Associated Blue Team Outcomes
Redundant camera system with lesser performance	Visual sensors	H1, H2, H4	1.2, 2.1, 2.2
System parameter assurance	Control station, control station based log of obstacles, classification algorithms	H1, H2, H3, H4	1.2, 2.1, 2.2
Triple redundant acoustic sensors for increased confidence in proximity measurement	Characterization sensors	H1, H4	1.2, 2.1, 2.2
Diversely redundant, hopping command sending capability	Control station	H1, H2, H3	1.1, 1.2, 1.4, 3.2
Two-factor command authorization	Obstacle	H1, H2	1.1, 1.4
Operational consistency checking for obstacle feedback	Obstacle	H1	1.4, 3.1
Backup communication network	Network	H1, H3	1.2, 2.1, 2.2, 3.2

Table 14: Resilience solutions mapped to their locations for implementation and mitigated hazardous conditions.

sensor giving bad measurements to be voted out. However, despite this solution's apparent effect on the accuracy of proximity measurements, the operation of Silverfish is not majorly affected by an increase in precision from its acoustic sensors.

Silverfish relies heavily on the visual surveillance from the cameras monitoring the denied area. Furthermore, the operator uses his or her own judgment for target identification and characterization– which is in itself an effective resiliency measure. The ability of the operator to maneuver within the denied area to make "eyes-on" assessments of agents within the denied area reduces the impact of increasing noise or bias to the sensors used for surveillance. However, it should be noted that, in the feasible near-future scenario where target identification and classification is automated, the impact of solutions that mitigate the introduction of noise or bias to the system's sensors increases significantly. Table 15 presents an overall impact rating for each resiliency solution based on the three factors mentioned above along with a rationale for the rating of the solution's effect on the system.

Following the classification of each resiliency solution's impact, the likelihood of each solution's outcomes to be mitigated is assessed. This likelihood measure is based on the number and severity of the changes made to the behavior simulation to achieve adverse outcomes. Table 16 presents

Solution	# of outcomes addressed	Priority of outcomes	Solution Effect	Overall Impact rating
Redundant camera system with lesser performance	3	1.67	Effect diminished due to ability of operator to visually confirm	Medium
System parameter assurance	4	1.67	Confidence in accuracy of state estimations increased, changes easily detected	High
Triple redundant acoustic sensors for increased confidence in proximity measurement	2	1.67	Minimal effect due to ability of operator to visually confirm	Low
Diversely redundant, hopping command sending capability	3	2	Assurance that commands are not altered within the control station is enhanced	High
Two-factor command authorization	2	1	Assurance that obstacles only perform legitimate commands is enhanced	Medium
Operational consistency checking for obstacle feedback	1	2	Assurance that obstacle is reporting the correct feedback enhanced	Low
Backup communication network	2	2	Backup network allows mission to continue if primary network goes down, huge impact on ability to complete mission	High

Table 15: Impact ratings for identified resiliency solutions.

each resiliency solution mapped to these two factors.

Now that each solution's impact and likelihood ratings have been recorded, each solution can be classified into a "risk" prioritization category as described in Section 4.1. Table 17 presents each resiliency solution's impact and likelihood ratings along with its prioritization category.

Figure 23 presents this same information in the risk matrix figure from Section 4.1

Solution	# of changes to achieve adverse outcome	Severity of changes	Overall Likelihood Rating
Redundant camera system with lesser performance	2	Low- difficult to achieve needed amount of noise or bias to affect behavior	Low
System parameter assurance	3	High- simple changes drastically affect system behavior	Medium
Triple redundant acoustic sensors for increased confidence in proximity measurement	2	Low- difficult to achieve needed amount of noise or bias to affect behavior	Low
Diversely redundant, hopping command sending capability	3	Medium- simple changes to affect system behavior, but difficult to achieve	Medium
Two-factor command authorization	1	Medium- simple changes to affect system behavior, but difficult to achieve	High
Operational consistency checking for obstacle feedback	1	Medium- simple changes to affect system behavior, but difficult to achieve	High
Backup communication network	1	High- simple changes drastically affect system behavior	High

Table 16: Likelihood ratings for identified resiliency solutions.

As seen in the above table and figure, the STRAT identified a total of seven resilience strategies appropriate for the Silverfish system. Of these seven, four are recommended to receive high priority for consideration for implementation, one for medium priority, and the remaining two should receive low priority. The low priority strategies, triple redundant acoustic sensors and a backup camera system, received a low ranking due to the nature of the Silverfish system. Since Silverfish uses a human-in-the-loop for identifying agents within the denied area and determining which obstacle to fire (in the event that the agent is an enemy), the likelihood that noise or bias in the sensor feedback causes incorrect behavior is minimal. In the event that the operator is unable to identify a target using the sensors, he or she will likely resort to visual confirmation, which makes the system resilient against friendly fire incidents at the slight risk of enemy agents traversing the denied area before they can be positively identified and engaged.

The medium-priority solution, operational consistency checking for obstacle feedback, received its ranking due to the ease with which adverse outcomes could be achieved by altering the system's perception of the obstacles states. More specifically, the concern behind this resiliency solution involves the breakdown of mission function if the system believes that an obstacle has been fired when it has not. Such a scenario could enable an adversary to traverse the denied area unimpeded if there were a pathway created by obstacles thought to have been fired.

The high-priority solutions labelled D and E in Table 17 involve measures to ensure that the com-

	Solution	Impact Rating	Likelihood Rating	Prioritization Category
A.	Redundant camera system with lesser performance	Medium	Low	Low
B.	System parameter assurance	High	Medium	High
C.	Triple redundant acoustic sensors for increased confidence in proximity measurement	Low	Low	Low
D.	Diversely redundant, hopping command sending capability	High	Medium	High
E.	Two-factor command authorization	Medium	High	High
F.	Operational consistency checking for obstacle feedback	Low	High	Medium
G.	Backup communication network	High	High	High

Table 17: Resilience solutions classified in their prioritization categories.

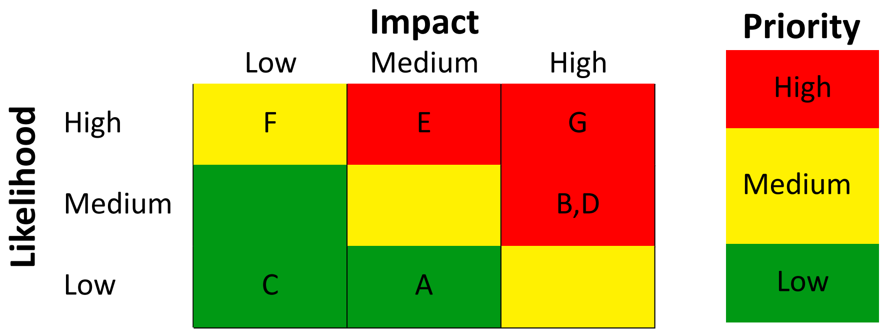


Figure 23: Resilience solutions mapped to their position in the risk matrix.

mands sent by a system component match the commands received by the intended recipient. It is clear how such mis-matches would result in unintended behavior. The solution labelled B in Table 17 involves ensuring that changes to the algorithms, decision models, or other parameters are detected and accounted for before any system processes are adversely affected.

Finally, the highest priority solution for consideration for implementation is the introduction of a backup communication network. Without a working network, the Silverfish system cannot complete its mission.

5.2.5 Comparison to CSRM Results

Comparing the results of the CSRM to the STRAT must first be qualified by the difference in goals between the two methodologies. First and foremost, the CSRM intends to develop cyber security requirements for a developing system. Such requirements include the incorporation of both resilience and traditional security solutions into the system design. STRAT, on the other hand focuses solely on the identification and evaluation of resiliency strategies. Consequently, the comparison of results between the two methodologies will be limited to the CSRM's recommendations for resiliency. Secondly, the CSRM leverages domain experts throughout the process, which provides excellent credibility in its results, but limits its applicability in practice due to scheduling and budget constraints. STRAT utilizes domain experts to a lesser degree in the hope that model-based evidence can provide a similar level of credibility.

The potential resilience strategies identified by the CSRM included resilient weapon control capabilities, diverse communications sub-systems, and resilient situational awareness capabilities. Based on the inputs from the Blue and Red Teams, the CSRM identified the system's communication subsystems (network) as the top priority area for resiliency. The resilient weapon control and situational awareness capabilities incorporated a variety of solutions such as diverse redundancy, confidence testing, and situational awareness introspection. The STRAT also identified the network as the top priority for resiliency, and specified that resiliency should be achieved through redundancy. The other resiliency strategies however, can be classified as sub-strategies of the weapon control and situational awareness categories identified in the CSRM.

In general, CSRM identified a broader selection of resiliency strategies than the STRAT. This could be a result of the abstraction of system hardware components in STRAT's definition of the control structure and system model. CSRM defined the existence hardware components explicitly in SysML representations, which likely aided the identification of resiliency strategies such as separating situational awareness information from weapon control functions.

The similarity of the top priority recommendations in the CSRM and STRAT suggests that STRAT is compatible with existing methodologies. Since STRAT does not involve as many domain experts as the CSRM, the STRAT could be a viable alternative technique for recommending a prioritized set of resiliency strategies. Both methodologies, however, are limited in that neither account for the cost of resiliency.

5.3 Application of CYBOK & Security Analyst Dashboard

We use the SysML models presented above with some extra design information added through the dashboard. This extra design information assists CYBOK to find applicable attack vectors. We note that a systems engineering analysis is already conducted and, therefore, it is not necessary to transfer the results to the dashboard. For these results we take the previous section as input to the attack vector analysis.

CYBOK and especially its UI consist of a useful tool to investigate and explore the attack vector space associated with a given system topology. This is because of their interactive nature between vulnerability data and the model, which is not present in the SysML framwork.

The evidence produced by the tools present in this report are also supported by previous efforts in the same realm [3]. Specifically, in the previous exercise stakeholders, blue team members, and red team members evaluated silverfish for critical subsystems and potential resilience or defensive solutions that might be applied before the system is deployed. The attack vector analysis supports

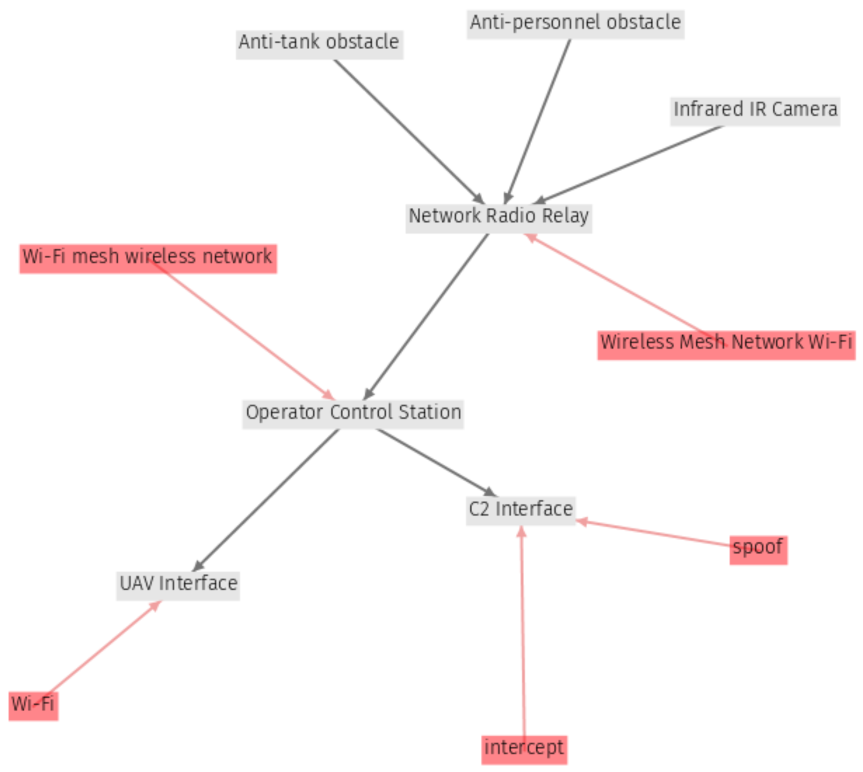


Figure 24: The attack surface of silverfish.

those findings by producing real attack information that could further provide assurance that the system is secure up to its operational needs.

Moreover, while this tool does not eradicate the need for consulting security professionals, it does allow for system designers to be better informed about potential attack vectors that should be discussed during those consultations. At the same, it provides an equal framework for both security analysts and systems engineers to discuss within the same language or framework and, therefore, it bridges the grap between the two.

5.3.1 Attack Surface

The first thing we look at is the automatically produced attack surface. This attack surface is constructed by looking at the *Entry Point* attribute in the model. If an attack vector is associated with the keywords in this specific attribute then the element; that is, vertex, becomes part of the attack surface. That means, that this element is very likely accessible externally by an attacker and if violated can be the reason for further spread—through other attack vectors—within the system topology.

Specific to silverfish we identified the following elements are part of the attack surface: UAV Interface, Operator Control Station, C2 Interface, Network Radio Relay. The violated attributes associated with these components are shown in red and are: Wi-Fi, Wi-Fi mesh wireless network, Intercept and Spoof (Figure 24). It is apparent, that all those elements are the communication

Table 18: A fragment of the results produced by CYBOK and picked by the security analyst through the Security Analyst Dashboard.

Network Radio Relay	
CAPEC-158	Sniffing Network Traffic
CWE-311	Missing Encryption of Sensistive Data
CWE-319	Cleartext Transmission of Sensitive Information
Operator Control Station	
CAPEC-10	Buffer Overflow via Environment Variables
CWE-120	Buffer Copy without Checking Size of input ('Classic Buffer Overflow')
CWE-231	Improper Handling of Extra Values
CWE-993	SFP Secondary Cluster: Incorrect Input Handling
UAV Interface	
CAPEC-604	Wi-Fi Jamming
CAPEC-202	Create Malicious Client
CWE-602	Client-Side Enforcement of Server-Side Security
CWE-254	7PK - Security Features
C2 Interface	
CWE-311	Missing Encryption of Sensistive Data
CWE-319	Cleartext Transmission of Sensitive Information
Infrared IR Camera	
CAPEC-13	Subverting Environment Variable Values
CWE-15	External Control of System or Configuration Setting
CWE-994	SFP Secondary Cluster: Tainted input to variable

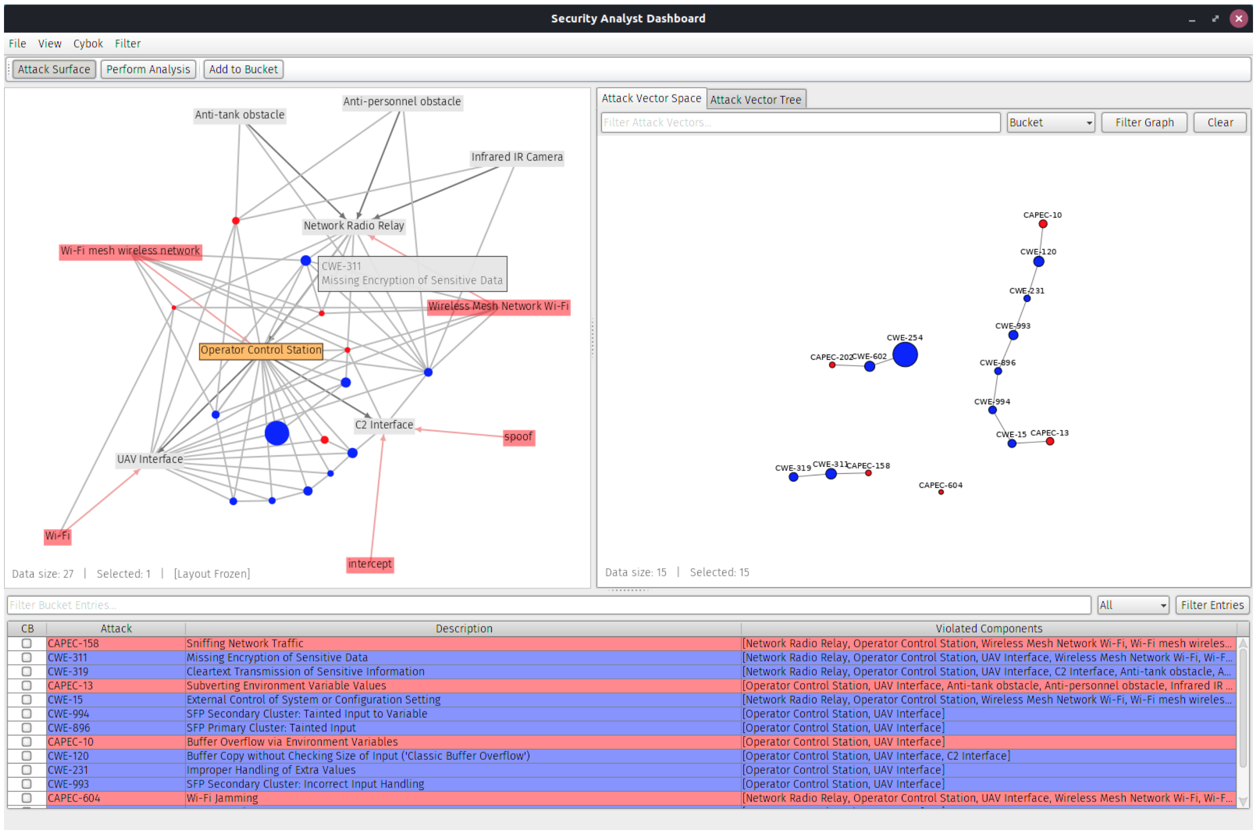
devices between subsystems.

5.3.2 Relevant Attack Vectors

To further understand how exploits can cause system function violations it is necessary to examine the attack vector space of the system topology; that is, the attacks produced by CYBOK to be relevant to it. To do so a security analyst has to filter through the attacks and examine which ones are: (1) truly applicable to the system, (2) have a high likelihood of being used against the system, and (3) be successful in violating subsystem functions.

The subset of attack vectors filtered by a security analyst using the Security Analyst Dashboard is significantly lower than the starting set (i.e., CAPEC, CWE, CVE) but also from the CYBOK set. CYBOK acts as the first filter, finding only relevant attack vectors from the databases based on the system model. Then, a further circumspection from a security analyst is necessary to find the truly applicable but also important attacks. The process is semiautomatic in that way.

The security analyst picks those results by using extra attributes of each component and using the automatically produced attack surface as an initial point of analysis. The individual attacks are found by using the tree view pane's filter function by component name or other information present in the datasets or the model. For example, the C2 Interface included the attribute



CB	Attack	Description	Violated Components
☐	CAPEC-158	Sniffing Network Traffic	[Network Radio Relay, Operator Control Station, Wireless Mesh Network Wi-Fi, Wi-Fi mesh wireles...
☐	CWE-311	Missing Encryption of Sensitive Data	[Network Radio Relay, Operator Control Station, UAV Interface, Wireless Mesh Network Wi-Fi, Wi-F...
☐	CWE-319	Cleartext Transmission of Sensitive Information	[Network Radio Relay, Operator Control Station, UAV Interface, C2 Interface, Anti-tank obstacle, A...
☐	CAPEC-13	Subverting Environment Variable Values	[Operator Control Station, UAV Interface, Anti-tank obstacle, Anti-personnel obstacle, Infrared IR ...
☐	CWE-15	External Control of System or Configuration Setting	[Network Radio Relay, Operator Control Station, Wireless Mesh Network Wi-Fi, Wi-Fi mesh wireles...
☐	CWE-994	SFP Secondary Cluster: Tainted Input to Variable	[Operator Control Station, UAV Interface]
☐	CWE-896	SFP Primary Cluster: Tainted Input	[Operator Control Station, UAV Interface]
☐	CAPEC-10	Buffer Overflow via Environment Variables	[Operator Control Station, UAV Interface]
☐	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	[Operator Control Station, UAV Interface, C2 Interface]
☐	CWE-231	Improper Handling of Extra Values	[Operator Control Station, UAV Interface]
☐	CWE-993	SFP Secondary Cluster: Incorrect Input Handling	[Operator Control Station, UAV Interface]
☐	CAPEC-604	Wi-Fi Jamming	[Network Radio Relay, Operator Control Station, UAV Interface, Wireless Mesh Network Wi-Fi, Wi-F...

Figure 25: The projected attacks over the system topology, the set of attack vectors picked by the security analyst in graph form, and the bucket containing the same attack vectors in textual form.

`communication_node: cleartext` which suggests that the security analyst should look for attacks related to cleartext or encryption. UAV Interface is part of the attack surface because it used Wi-Fi and attacks on Wi-Fi are possible. Finding those attacks is important to inform about what defenses or mitigations. Infrared IR Camera is a device that the security analyst knows that it uses external environment input data, so I searched for attacks related to that. The resulting set of attack vectors is only the crucial set that the security analyst reports to the rest of the stakeholders (Table 18). Another view of this smaller set can be to project it over the system topology, which further informs at the critical subsystems of the whole system (Figure 25).

Because all attack surface elements are part of the communications network the attack vector analysis agrees with the CSRM and STRAT analyses presented in previous sections. Additional to those results, however, the attack vector analysis reveals the potential for further spread within the system in the event that those attack surfaces are accessed by an attacker; that is, in the event that the resilience solutions are insufficient. In that case, further barriers might be necessary—based on the stakeholders's needs—within the system structure itself. The specific attack vectors reported by CYBOK and the Security Analyst Dashboard can guide such defense strategies.

6 Conclusions and Potential Future Research Directions

Outcomes this year include developing a deeper understanding of open source databases of historical cyber attacks (e.g., CAPEC, CWE, CERT, and CVE), as well as defining and developing 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) development of the STRAT toolset to support CSRM and dynamic assessment of attack consequence, (2) use of several different NLP/querying techniques to characterize relationships between attack classes in CAPEC, CWE, and CVE; (3) development of the Security Analyst Dashboard. 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. The Dashboard is designed to work within CYBOK (though CYBOK may be used independently of the dashboard); for example, the dashboard uses the automated recommender system that underpins CYBOK to provide analysts with the capability to directly query specific entries in CAPEC, CVE, and CWE.

For future research efforts, consideration should be placed on how to accelerate the transition of research results into practice. The recommended path for such a transition would be to engage in a case study that features collaboration with one or more tool vendors as a basis for addressing the technical issues related to the integration of the newly derived tools with existing SysML-based MBSE tool sets. This case study approach to accelerating transition into practice would require selection of a toolset for initial evaluations.

Currently CYBOK is engineered to employ MITRE Corporation's CAPEC and associated databases. Future efforts might focus on the identification of additional data requirements that would enhance support for evolving cyber resilience risk assessments. The motivation for such an effort is the expectation that future cyber attacks will increase the need to address cyber-physical system features and system-of-system integration features, thereby requiring different sets of information and associated schema than are employed in the current MITRE data sets. Additionally, there is a need for a risk scoring system that combines the likelihoods from CYBOK with systems behavior derived from the dynamic, finite-state analyses from STRAT, with the promise that higher fidelity assessments of cyber attack consequences (based on system behavior rather than structure), and consequently improved risk analyses, could be produced.

References

[1] C. H. Fleming. Systems theory and a drive towards model-based safety analysis. In *2017 Annual IEEE International Systems Conference (SysCon)*. IEEE, 2017.

[2] P. M. Beach, R. F. Mills, B. C. Burfeind, B. T. Langhals, and L. O. Mailloux. A stamp-based approach to developing quantifiable measures of resilience. In *Proceedings of the International Conference on Embedded Systems, Cyber-physical Systems, and Applications (ESCS)*, 2018.

[3] Barry Horowitz, Peter Beling, Cody Fleming, Stephen Adams, Bryan Carter, Tim Sherburne, Carl Elks, Georgios Bakirtzis, Forrest Shull, and Nancy R Mead. Cyber security requirements methodology. Technical report, Stevens Institute of Technology Hoboken United States, 2018.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of machine Learning research*, 2003.

[5] S. Barnum. Common attack pattern enumeration and classification (CAPEC) schema description. *Cigital Inc, http://capec. mitre. org/documents/documentation/CAPEC_Schema_Description_v1*, 3, 2008.

[6] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 1951.

[7] S. Adams, B. Carter, C. Fleming, and P. A. Beling. Selecting system specific cybersecurity attack patterns using topic modeling. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018.

[8] X. Yuan, E. B. Nuakoh, J. S. Beal, and H. Yu. Retrieving relevant CAPEC attack patterns for secure software development. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*. ACM, 2014.

[9] I. Kotenko and E. Doynikova. The CAPEC based generator of attack scenarios for network security evaluation. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on*. IEEE, 2015.

[10] X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006.

[11] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE international conference on Software Engineering*. ACM, 2010.

[12] B. Lu, M. Ott, C. Cardie, and B. K. Tsou. Multi-aspect sentiment analysis with topic models. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2011.

[13] L. Na, X. Peng, L. Ying, T. Xiao-Jun, W. Hai-Wen, and L. Ming-Xia. A topic approach to sentence ordering for multi-document summarization. In *Proceedins of the 2016 IEEE Trustcom/BigDataSE/I SPA*. IEEE, 2016.

[14] K. Aswani, A. Cronin, X. Liu, and H. Zhao. Topic modeling of SSH logs using latent Dirichlet allocation for the application in cyber security. In *Proceedings of the 2015 Systems and Information Engineering Design Symposium, SIEDS*. IEEE, 2015.

[15] F. Kolini and L. Janczewski. Clustering and topic modelling: A new approach for analysis of national cyber security strategies. In *Proceedings of PACIS 2017*, 2017.

[16] S. Neuhaus and T. Zimmermann. Security trend analysis with CVE topic models. In *Proceedings of the 2010 IEEE 21st international symposium on Software reliability engineering, ISSRE*. IEEE, 2010.

[17] Barry Horowitz, Peter Beling, Cody Fleming, Stephen Adams, Bryan Carter, Krishnamurthy Vemuru, Carl Elks, Tim Bakker, Kryzsztof Cios, Georgios Bakirtzis, et al. Security engineering fy17 systems aware cybersecurity. Technical report, Stevens Institute of Technology Hoboken United States, 2017.

[18] T. L. Griffiths, M. I. Jordan, J. B. Tenenbaum, and D. M. Blei. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, 2004.

[19] D. M. Blei and J. D. Lafferty. Correlated topic models. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*. MIT Press, 2005.

[20] J. D. Mcauliffe and D. M. Blei. Supervised topic models. In *Advances in neural information processing systems*, 2008.

[21] M. B. Christopher. *Pattern Recognition and Machine Learning.* Springer, 2006.

[22] C.-J. Kim and C. R. Nelson. *State-space models with regime switching: classical and Gibbs-sampling approaches with applications*. The MIT press, 1999.

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

[24] Martin F Porter. An algorithm for suffix stripping. *Program*, 1980.

[25] J. Cao, T. Xia, J. Li, Y. Zhang, and S. Tang. A density-based method for adaptive LDA model selection. *Neurocomputing*, 2009.

[26] R. Arun, V. Suresh, C. E. V. Madhavan, and M. N. N. Murthy. On finding the natural number of topics with latent Dirichlet allocation: Some observations. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010.

[27] R. Deveaud, E. SanJuan, and P. Bellot. Accurate and effective latent concept modeling for ad hoc information retrieval. *Document numérique*, 2014.

[28] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou. Exploiting Wikipedia as external knowledge for document clustering. In *Proceedings of the 15th International Conference on Knowledge discovery and data mining, SIGKDD*. ACM, 2009.

[29] O. Jin, N. N. Liu, K. Zhao, Y. Yu, and Q. Yang. Transferring topical knowledge from auxiliary long texts for short text clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011.

[30] D. Andrzejewski, X. Zhu, M. Craven, and B. Recht. A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2011.

[31] Z. Chen, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Leveraging multi-domain prior knowledge in topic models. In *Proceedings of the 2013 International Joint Conferences on Artificial Intelligence, IJCAI*, 2013.

[32] W. Young and N. Leveson. Systems thinking for safety and security. In *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013.

[33] G. Bakirtzis, B. T. Carter, C. H. Fleming, and C. R. Elks. Mission aware: Evidence-based, mission-centric cybersecurity analysis. *arXiv preprint arXiv:1712.01448*, 2017.

[34] H. Derksen and J. Weyman. Quiver representations. *Notices of the AMS*, 2005.

[35] R. A. Jones and B. Horowitz. A system-aware cyber security architecture. *Systems Engineering*, 2012.

[36] P. R. Garvey and Z. F. Lansdowne. Risk matrix: an approach for identifying, assessing, and ranking program risks. *Air Force Journal of Logistics*, 1998.

[37] G. Bakirtzis, B. T. Carter, C. R. Elks, and C. H. Fleming. A model-based approach to security analysis for cyber-physical systems. In *Proceedings of the 2018 Annual IEEE International Systems Conference, SysCon 2018*, 2018.

[38] B. T. Carter, G. Bakirtzis, C. R. Elks, and C. H. Fleming. A systems approach for eliciting mission-centric security requirements. In *Proceedings of the 2018 Annual IEEE International Systems Conference, SysCon 2018*, 2018.

[39] G. Bakirtzis, B. J. Simon, C. H. Fleming, and C. R. Elks. Looking for a black cat in a dark room: Security visualization for cyber-physical system design and analysis. *arXiv preprint arXiv:1808.08081*, 2018.