Resiliency Analysis of Mission-Critical System of Systems Using Formal Methods

Mahmoud Abdelgawad(✉) and Indrakshi Ray

Department of Computer Science, Colorado State University, Fort Collins, CO 80523, USA
{m.abdelgawad,indrakshi.ray}@colostate.edu

Abstract. A System of Systems (SoS) is a formation of heterogeneous systems that work together effectively to accomplish critical missions. These mission-critical SoS ensure the safety and security of any nation. Attacks on mission-critical SoS can have devastating outcomes. We need to design missions that are resilient to attacks. SoS engineering must specify, analyze, and understand where adverse events are possible and how to mitigate them while a mission-critical SoS is deployed. This work uses an end-to-end methodology for analyzing the resiliency of mission-critical SoS. The methodology starts with representing SoS mission as a workflow. It then converts the workflow into formal representation using Coloured Petri Nets (CPN). Threat models are extracted from the mission specification, which are used to construct the CPN representations of the attacks. The mission specifications and the attacks are composed to analyze the mission resiliency. The analysis identifies if the mission succeeds, fails, and is incomplete. We apply the methodology to an SoS consisting of a military vehicle and route-reconnaissance drones working together to monitor a national border and respond immediately to any physical threats. The result demonstrates how to restrict the mission to improve the resiliency of SoS formation. Such methodology is important for the early design stage of resilient mission-critical SoS.

Keywords: System of Systems (SoS) · Mission-critical Systems · Resiliency Analysis · Workflow · Formal Methods · Coloured Petri Nets (CPN)

1 Introduction

A System of Systems (SoS) is a formation of heterogeneous systems that interact for a global goal [4]. These systems operate independently, but none can accomplish the goal on its own; thus, they work together towards broader objectives. SoS characteristics include being independently managed, geographically distributed, evolving to meet mission needs, emerging from a combination of behaviors and properties of the system elements, and being influenced by a stimulus environment [2,4]. SoS has become common in technology domains, including transportation, water and energy resources, and defense. The SoS standardization [7] also considers Smart Grids, Smart Cities, and Internet-of-things (IoT) as part of one or more SoS. This paper is scoped to the SoS domains in defense where an operational environment, the objectives are established based on critical missions [5,11,19]. Most military SoS formations, for instance, ships,

A. L. Ferrara and R. Krishnan (Eds.): DBSec 2024, LNCS 14901, pp. 153–170, 2024.
https://doi.org/10.1007/978-3-031-65172-4_10

aircraft, satellites, and ground vehicles, are equipped with independent system elements (i.e., sensors, weapons, communications) that must fulfill mission-critical objectives. These mission-critical systems are defined as systems whose one system element failure may significantly impact the mission-related functions [6, 15]. Hence, resilience analysis of the mission-critical SoS is essential. The resiliency of mission-critical SoS is identical to cyber survivability defined for warfighter systems [17]. Mission-critical SoS must fulfill survivability requirements to continue a mission in the face of attacks. Therefore, SoS system engineering must specify and analyze a mission before deployment to evaluate its resilience and gauge what failures can be tolerated.

A mission can be described as a workflow consisting of various tasks executed by subjects and connected via different types of control-flow operators [20]. Mission also has a set of objectives. If all the objectives are satisfied, then the mission has *succeeded*. If some, but not all, objectives are satisfied, the mission is *incomplete*. When no objectives have been satisfied, the mission has *failed*. Many types of workflow resilience are defined: static, decremental, and dynamic resilience [18]. Static resiliency refers to a situation in which subjects become unavailable before the workflow executes, and no subjects may become available during the execution. Decremental resiliency expresses a situation where subjects become unavailable before or during the execution of the workflow, and no previously unavailable subjects may become available during execution. This work addresses the decremental resilience. Analyzing the workflow resiliency manually is error-prone and labor-intensive. Towards this end, there is a need for automated analysis of a mission and evaluating potential attacks and their effects. We use Coloured Petri Nets (CPN) [10] for the purpose of analysis. CPN is suitable for modeling process interaction and provides primitives for defining data types (i.e., colors). It is also associated with a high-level programming language (i.e., CPN-ML [8]) and Integrated Development Environment (i.e., CPN Tools [16]) that are practical to simulate processes interaction, manipulate data values, and verify state transitions of systems.

We designed a methodology that analyzes various aspects of mission resiliency for mission-critical SoS. The methodology leverages the workflow definition to describe a mission. It also utilizes a set of transformation rules [1] to convert the workflow to CPN. The methodology starts with specifying a mission and formally representing it as a workflow. The mission workflow is converted into the formal CPN specification which we refer to as CPN mission. The methodology systematically identifies threat models from the mission specification. The threat models are converted into formal CPN specifications, which we refer to as CPN attacks. These CPN attacks are used to demonstrate various attack scenarios. The CPN attacks are composed with the CPN mission, and the resulting system analyzed. It analyzes the mission's resiliency as it identifies when the mission succeeds, fails, and is incomplete.

The methodology is applied to an SoS formation consisting of a military vehicle required to monitor a national border and respond immediately to any threats. It also includes two unmanned aerial vehicles (i.e., drones) that perform surveillance and reconnaissance for the mission. A military base exchanges information between the military vehicle and drones to ensure a mission's success. The SoS mission decremental resiliency defined in [18] is analyzed. The application shows that the methodology is practical for mission analysis and helps improve mission-critical SoS resilience.

2 SoS Mission Description

This section describes a mission-critical SoS as a formation of a military vehicle and two rotary-winged drones. The military vehicle and drones exchange data through a Line Of Sight (LOS) communication. The military vehicle also exchanges data with an operation center site (i.e., a military base) through a Beyond Line Of Sight (BLOS) via satellite communication.

The mission requires the military vehicle to monitor the national border between the official ports of entry and respond to any threat. We assume the mission will occur in a region of interest with gravel roads and rough terrain. Human elements, the military vehicle drivers (e.g., soldiers), usually plan several routes to reach a Point of Interest (POI). In many cases, however, it is challenging to anticipate route hazards they may encounter (e.g., driving down a steep hill). Thus, drones play the role of surveillance and reconnaissance for the military vehicle to keep a safe distance from dangerous situations. The drones fly 5 mi ahead of the military vehicle at low altitudes using Global Positioning System (GPS) to inform locations. The drones accelerate and decelerate to keep a 5-mile distance from the military vehicle. They are also equipped with cameras and infrared sensors for transmitting day and night visions. If the drones fly during the daytime, they will use the cameras; if it is nighttime, they will use the infrared sensors. The drones keep flying as long as the battery is enough to return to the military vehicle for recharging. We assume the battery level contains 100 units, enough to fly for 4 h. If the drone is 5 mi from the military vehicle, it needs 2 battery units to return to the vehicle for recharging. The military vehicle is estimated to take 30–45 minutes to reach the POI if the human elements drive at average speed (60–70 miles per hour). The military vehicle usually reaches the POI, evaluates the situation, and returns to the starting point (i.e., station) without engagement. If the situation at the POI is evaluated for engagement, the human elements must receive instruction from the military base to engage (e.g., open fire against a threat). The mission requires the military vehicle to be fueled with a full petrol tank (i.e., gasoline) and connected to the military base. Actuating the cameras and infrared sensors, each takes one battery unit. Before flying, the drone's status is checked (LOS communication is up, cameras and infrared sensors are on, and the battery is fully charged). The military vehicle's status (full gasoline tank and connected to the military base) is also checked during deployment to accomplish a mission. The mission is successful if the military vehicle and drones reach the POI in time, report to the military base about the POI situation, and return to the station safely. The mission is considered incomplete if drones cannot collect data but can fly back to the vehicle, and the vehicle reaches the POI but cannot report to the military base. The mission fails when the drones cannot return to the vehicle (e.g., lost), the vehicle cannot reach the POI, or the vehicle or the drones do not return to the station.

Now, we need to analyze whether it is possible for the mission to succeed and in which attack scenarios the mission fails. An attack scenario changes the military vehicle and drones' attributes (i.e., mutable attributes that can be manipulated) and renders this SoS formation unable to perform a task. For instance, an attack scenario defuses the drones' cameras. It degrades its scanning capability, leaving the military vehicle unable to figure out the situation ahead, which may cause a mission to fail. Another attack scenario disrupts the BLOS communication, rendering the military vehicle disconnected from the military base and unable to report the threat situation, which may cause an incomplete mission. The methodology systematically covers possible attack scenarios that cause SoS degradation, and each attack scenario must be analyzed to determine if mission can be successful, failed, or incomplete with respect to the attack. A mission is resilient to an attack scenario if, after the attack occurs, it still meets its objectives.

3 Resiliency Analysis Methodology for Critical-Mission SoS

As illustrated in Fig. 1, the methodology consists of 7 steps, described below.

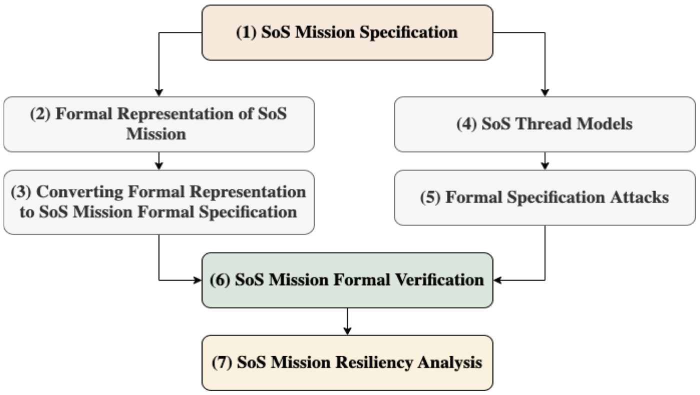
(1) SoS Mission Specification
(2) Formal Representation of SoS Mission
(4) SoS Thread Models
(3) Converting Formal Representation to SoS Mission Formal Specification
(5) Formal Specification Attacks
(6) SoS Mission Formal Verification
(7) SoS Mission Resiliency Analysis

Fig. 1. Methodology for Mission Resiliency Analysis

Step 1 - SoS Mission Specification: It defines the mission *subjects* as active entities that execute tasks. Each *subject* has a type and a set of variables corresponding to its *attributes*. Some attributes of a subject (such as id) have values that cannot be changed – these are referred to as *immutable attributes*. All other attributes whose values can be changed are referred to as *mutable attributes*. The values of the attributes constitute a subject state. The state of a subject determines whether it can execute a given task. Step 1 also defines a set of *tasks*, which are atomic units of the mission. The mission starts with an initial task and ends with a final task. A mission includes a set of intermediate tasks between the initial and final tasks that may be grouped into sub-sets of tasks, referred to as sub-workflow in the formal representation. A mission also has a set of *objectives* to be accomplished.

Step 2 - Formal Representation of SoS Mission: This step represents the mission as a workflow. It decomposes the mission into many sub-workflows and connects them into a main workflow to represent the entire mission. It then instantiates the workflow by initializing each subject and its attributes, assigning each subject to tasks, and assigning initial conditions.

Step 3 - Converting Formal Representation to SoS Mission Formal Specification: The mission workflow is transformed into a formal specification for verifying the correctness and analyzing its resiliency.

Step 4 - SoS Threat Models: The threat models are derived from the mission specification. In this work, we focus on threats that may change the values of a subject's mutable attributes. When a subject's attribute value changes, it impacts the mission.

Step 5 - Formal Specification of Attacks: Each attack changes the mutable values of a subject. An attack requires preconditions to be satisfied so that it can happen. The effect of the attack is described as a post condition. Thus, each attack is specified with a precondition and post condition.

Step 6 - SoS Mission Formal Analysis: This step composes the Mission CPN with the Attack CPN. The Attack CPN can be attached only to the places in the Mission CPN where the Attack CPN's preconditions are satisfied. The state space and state transitions are verified and analyzed for each attack scenario attached to the mission formal specification.

Step 7 - SoS Mission Resiliency Analysis: SoS mission resiliency can be analyzed, including decremental and dynamic resiliency [18]. The resiliency analysis highlights constraints that must be enforced to make the mission more resilient.

4 SoS Mission Specification to Formal Representation

This section applies the first two steps of the methodology to the SoS example described in Sect. 2 to translate it into formal CPN specification.

4.1 SoS Mission Specification

We manually analyze the mission description to define the subjects, their attributes, tasks, and mission objectives. Only mutable attributes are considered. The SoS mission specification is summarized as follows:

- set of subjects = $\{vehicle, [drones]\}$
- set of *vehicle* attributes = $\{location, GPS, speed, fuel, LOS, BLOS\}$
- set of *drone* attributes = $\{location, GPS, fly, acceleration, battery, camera,$ $IR\ sensor, LOS\}$

The tasks are:

1. check the *vehicle* status: *fuel* is full, LOS_v communication is up, and *BLOS* communication is up,
2. check the $[drones]$ status: *battery* is set to 100 units, *camera* is on, *IR sensor* is on, and *LOS* communication is up,
3. drive the *vehicle* toward the POI with an average speed of 60–70 miles per hour,
4. fly $[drones]$ toward the POI and keep 5 mi distance from the *vehicle*,
5. if it is daytime, each *drone* uses *camera* to scan the route and the POI area and provide day vision to the *vehicle*,
6. if it is nighttime, each *drone* uses *IR sensor* to scan the route and the POI area and provide night vision to the *vehicle*,
7. at the POI area, evaluate the situation, and report the military base in real-time,
8. if the threat requires engagement, the human element must receive instruction from the military base to engage, and

9. the *vehicle* and [*drones*] return to the station with data collected and evaluation reported.

The objectives = {*move toward POI*, *collect data*, *evaluate the situation*, *respond to threads*, *return to the station*}. Notice that we italicized the mission elements, which are used to formalize the SoS mission representation.

4.2 Formal Representation of the SoS Mission

We use the workflow defined in [1] to represent a mission. A workflow consists of tasks connected through operators. The syntax of the workflow is defined as follows.

Definition 1 (Workflow). *A workflow is defined recursively as:*

$$W = t_i \otimes (t|W_1 \otimes W_2|W_1\#W_2|W_1\&W_2|if\{C\}\,W_1\;else\,W_2|\;while\{C\}\{W_1\}) \otimes t_f$$

where sequence, exclusive choice, and, conditioning, and iteration operator

- *t is a subject-defined atomic task.*
- *t_i and t_f are unique initial and final tasks, respectively.*
- *$\otimes$ denotes the sequence operator. $W_1 \otimes W_2$ specifies W_2 is executed after W_1 completes.*
- *# denotes the exclusive choice operator. $W_1\#W_2$ specifies that either W_1 executes or W_2 executes but not both.*
- *& denotes the and operator. $W_1\&W_2$ specifies that both W_1 and W_2 must finish executing before the next task can start.*
- *$if\{C\}\,W_1$ else W_2 denotes the conditioning operator. C is a Boolean-valued expression. Either W_1 or W_2 execute based on the result of evaluating C but not both.*
- *$while\{C\}\{W_1\}$ denotes iteration operator. If C evaluates to true, W_1 executes repeatedly until the expression C evaluates to false.*

A mission is defined as one or many subjects performing tasks to accomplish objectives.

Definition 2 (Mission). *A mission is defined as 5-tuple $\mathcal{M} = (W, \mathcal{S}, \mathcal{ST}, I, O)$ where W is the workflow corresponding to the mission, $\mathcal{S}$ is a set of subjects, $\mathcal{ST} \subseteq S \times Tasks\,(W)$ is the set of subject to task assignments, I is the set initial conditions, and O is the set of mission objectives. The conditions and objectives are expressed in predicate logic.*

The SoS mission is divided into three sub-workflows. *Deployment* sub-workflow comprises check status and deploy tasks. *Moving-to-POI* sub-workflow consists of drive the vehicle, fly drones, keep 5-miles distance ahead, and scan routes tasks. *At-POI* sub-workflow includes evaluate situations, report to the base, and engaging tasks. The *Deployment*, *Moving-to-POI*, and *At-POI* sub-workflows respectively execute sequentially. These three sub-workflows connect to the main workflow that starts with the initial and final tasks. Each sub-workflow forms a set of tasks managed by operators. The operators are also used to connect sub-workflows to the main workflow. The complete workflow of the SoS mission is described as follows:

$$W = init \otimes \{check\_status\,(vehicle) \otimes check\_status\,([drones]) \otimes deploy \otimes$$
$$\{if\,(daytime)\,\{turn\_on\,([cameras])\,else\{turn\_on\,([IR\ sensors])\}\} \otimes fly\,([drones]) \otimes drive\_vehicle\} \otimes$$

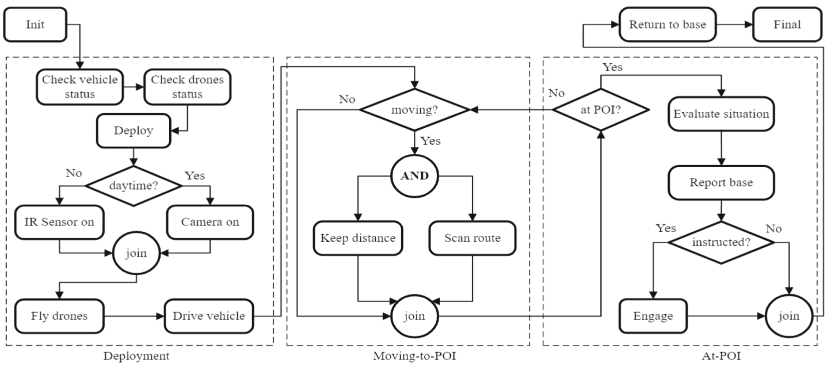

Fig. 2. SoS Mission Workflow

$while\,(moving)\,\{keep\_distance\ \&\ scan\_route\}\} \otimes if\,(locatio = POI)\,\{evaluate\_situation \otimes report\_base \otimes if\,(instruction)\,\{engage\}\} \otimes return\_to\_base \otimes final$

Figure 2 illustrates the workflow diagram of the SoS mission. The dotted boxes represents the sub-workflows, *Deployment*, *Moving-to-POI*, and *At-POI*. The *return to base* task can be part of the main workflow since it does not represent any sub-workflow. Visualizing the workflow helps ensure the correctness of the workflow transformation to the CPN; hence, it is practical to draw the workflow.

Now, we instantiate the workflow. We initialize each subject and its attributes, assign each subject to a set of tasks, and set initial conditions for the subjects and objectives. There are two subject types *Vehicle* and *Drone*, and are instantiated as $\mathcal{S} = \{vehicle : Vehicle;\ drones[\,] : Drone\}$. The subject type *Vehicle* has a set of attributes as $Vehicle.attributes = \{location : string;\ GPS : coordinate;\ speed \in \{0..240\};\ fuel \in \{0..80\};\ LOS : bool;\ BLOS : bool;\ evaluation\_reported : bool\}$. The subject type *Drone* has a set of attributes as $Drone.attributes = \{location : string;\ GPS : coordinate;\ fly : bool;\ acceleration \in \{1..10\};\ battery \in \{1..100\};\ camera : bool;\ IR\_sensor : bool;\ LOS : bool;\ data\_collected : bool\}$. The subjects *vehicle* and $[drones]$ are assigned to tasks as $\mathcal{ST} = \{((vehicle, [drones]),\ t) \mid t \in Tasks(W)\}$.
Let I be a predicate logic formula giving the initialization conditions as:
$I = \exists\, s : S \mid (type = Vehicle) \wedge (location = \text{“}station\text{”}) \wedge (GPS = [34.0489, -111.0937]) \wedge (speed = 0) \wedge (fuel = 80) \wedge (LOS = true) \wedge (BLOS = true) \wedge (evaluation\_reported = false) \wedge (type = Drone) \wedge (location = \text{“}station\text{”}) \wedge (GPS = [34.0489, -111.0937]) \wedge (fly = false) \wedge (acceleration = 0) \wedge (battery = 100) \wedge (camera = false) \wedge (IR\_sensor = false) \wedge (data\_collected = false)$.
Let O be a predicate logic formula that defines the mission objectives as follows:
$O = \exists\, s : S \mid (type = Vehicle) \wedge (type = Drone) \wedge (location = \text{“}station\text{”}) \wedge (evaluation\_reported = true) \wedge (data\_collected = true)$. We positioned the initial GPS coordinates for demonstration as [34.0489, -111.0937] and ranged the acceleration from 1 to 10. The GPS coordinate (i.e., Decimal Degrees (DD format) as [latitude, longitude]) is used to calculate the distance between the vehicle and the drones, and the acceleration is used to slow and speed up the drones to keep the 5-mile distance. As a result, the SoS mission specification is described as: $\mathcal{M}_{SoS} = (W, \mathcal{S}, \mathcal{ST}, I, O)$

4.3 Converting Formal Representation to SoS Mission Formal Specification

We consider CPN as a formal model that is practical to express mission state transitions due to CPN providing color sets, a flexible way to define data types to represent various types of subjects [9,10]. The CPN is formally defined as:

Definition 3 (Coloured Petri Nets (CPN)). *CPN is defined as 9-tuple $CPN = (P,T,A,\Sigma,V,C,G,E,I)$, where P, T, A, Σ, and V, are sets of places, transitions, arcs, colors, and variables, respectively. C, G, E, and I are functions that assign colors to places, guard expressions to transitions, arc expressions to arcs, and tokens at initialization expression, respectively.*

We use transformation rules defined in [1] to convert the mission workflow tuple into a CPN tuple as:

$$\mathcal{M} = (W,\mathcal{S},\mathcal{ST},I,O) \xrightarrow[Rules]{Trans.} CPN = (P,T,A,\Sigma,V,C,G,E,I)$$

The transformation rules are briefly explained through the SoS mission workflow:

Rule 1: Convert each task into a CPN transition.

Rule 2: Create input and output places for each transition, and create input and output arcs to pair the input place to the transition and the transition to the output place.

Rule 3: Produce a color set for each subject. The color set is a CPN datatype structure such as Product, Record, List, and Union [3]. The CPN color set combines primitive types such as *Bool*, *Int*, *Real*, and *String*. It also can be a compound of primitive types and color sets. The color sets represent subject attributes. The color set that represents the Vehicle is declared as:
$color: \ Vehicle = \ \textbf{record}\ \{location: String;\ GPS: [Real, Real];\ speed: Int \in \{1..240\};\ fuel: Int \in \{1..80\};\ LOS: Bool;\ BLOS: Bool;\ evaluation\_reported: Bool\}$
The color set that expresses the Drone is declared as:
$color: \ Drone = \ \textbf{record}\ \{location: String;\ GPS: [Real, Real];\ fly: Bool; acceleration: Int \in \{1..10\};\ battery: Int \in \{1..100\};\ camera: Bool; IR\_sensor: Bool;\ LOS: Bool;\ data\_collected: Bool\}$

Rule 4: Assign the color sets to the CPN places and arcs to declare their datatypes and declares a set of variables such that there is a variable for each color set. For instance, the declaration $var\ vehicle : Vehicle$ declares a variable $vehicle$ of type $Vehicle$. The expression $colset\ droneList = list\ Drone\ with\ 2$; declares a list of datatype $Drone$ with maximum two elements, and the declaration $var\ drones : droneList$ declares a variable $drones$ of type $dronelist$.

Rule 5: Assign a guard expression for each CPN transition. The guard expression must be evaluated true in order for a transition to be executed (i.e., fired). For instance, $fly(drone_1) = true$ is a guard expression assigned to the deployment transition. It ensures that the deployment transition fires only if the $drone_1.fly$ attribute is true.

Rule 6: Assign an arc expression for each input and output arc of a transition. The arc expressions evaluate tokens to pass in and out from transitions (i.e., bindings). When a token is evaluated true by an arc expression, it enables the transition that connects to be fired. For instance,

$if\,(at\_POI\,(vehicle) = true)\,\{vehicle\}\ else\ \{empty\}$ is an arc expression that ensures the *vehicle* token reaches the POI place so the token passes; otherwise, the POI place does not bind the next transition, and the next transition will not be executed.

Rule 7: Initialize variables with values and assign these variables to the CPN places as tokens. The initialization forms the initial conditions of the mission. The tokens are then manipulated in each CPN transition, generating the state transitions of the CPN. For instance, the initial conditions of the SoS mission are described as:

$vehicle = init\{(location = \text{“}station\text{”}) \wedge (GPS = [34.0489, -111.0937]) \wedge (speed = 0) \wedge (fuel = 80) \wedge (LOS = true) \wedge (BLOS = true) \wedge (evaluation\_reported = false)\}$

$drone_1, drone_2 = init\{(location = \text{“}station\text{”}) \wedge (GPS = [34.0489, -111.0937]) \wedge (fly = false) \wedge (acceleration = 0) \wedge (battery = 10) \wedge (camera = false) \wedge (IR\_sensor = false) \wedge (data\_collected = false)\}$.

Figure 3 illustrates the *CPN* model resulting from applying the conversion rules 1–7 to the SoS mission $\mathcal{M}_{SoS}$. We have explicitly omitted writing each arc expression into the *CPN* model to maintain readability. It shows the primary CPN mission, the SoS Mission main model, and three sub-CPN models: Deployment, Moving-to-POI, and At-POI. These three sub-CPNs are connected to the main CPN model through input and output ports, allowing the token to cross through the sub-CPN model and back to the main CPN. The small sub-CPNs, GPS attack, Battery, Camera, Sensor, LOS, and BLOS attack, depict CPN attack models, which are discussed next.

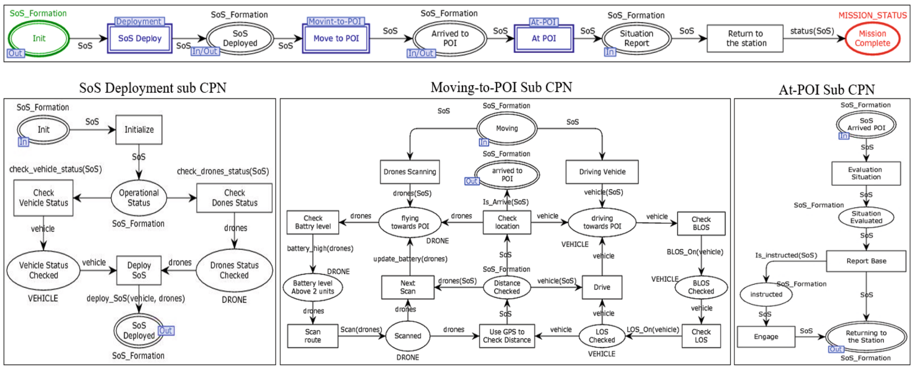

Fig. 3. CPN of the SoS Mission

5 Threat Model and Attacks Representation

This section describes the threat models and attack representations for the SoS mission.

5.1 SoS Threat Model

The threat model is derived directly from the mission description (Step 1). The vehicle and drones have mutable attributes that are attackable. We assume the drone can fly and accelerate are immutable since they are physical attributes (i.e., mechanical attributes), thus excluding them from the threat model. Other subjects with entirely immutable attributes, such as human elements and satellites, are also omitted from the threat model. We consider *GPS*, *LOS*, and *BLOS* to be mutable attributes of the vehicle. We also consider *GPS*, *battery*, *camera*, *IR_sensor*, *LOS*, are mutable attributes of the drones. Note that we focus on the communication and sensing capabilities of the military vehicle and drones since they are the best attackable attributes.

Table 1. Threat Model of the SoS Mission

Subject	Mutable Attribute	Attack Process	Impact on the Mission
Vehicle	*GPS*	Send out false signals to change the vehicle coordinates	The vehicle drives in the wrong direction, unable to reach the POI, leading to the mission to be *fail*. It may also cause the drones to hover away, unable to return, and the mission *fails*
	LOS	Disable the LOS communication	The vehicle loses communication with the drones, losing the route vision ahead; it may cause the mission to be *incomplete*
	BLOS	Disable the BLOS communication	The vehicle loses communication with the base, unable to receive instruction and unable to report the situation in real-time, leading the mission to be *failed*
Drone	*GPS*	Send out false signals to change the drones coordinates	The drones surveil incorrect route and collect useless data leading to the mission to be *incomplete*. They may also be unable to return to the vehicle and the mission *fails*
	battery	Consume one battery unit	It degrades the drones' ability to collect data and reduces flying time, leading to the mission to be *incomplete*; it might lead to mission *fails* if the drones do not have enough battery to return to the vehicle
	camera	Disable the camera	The drones are incapable of collecting data and cannot provide daytime vision to the vehicle, leading to the mission to be *incomplete*
	IR_sensor	Disable the camera	The drone is incapable of collecting data and cannot provide nighttime vision to the vehicle, leading to the mission to be *incomplete*
	LOS	Disable the LOS communication	The drone loses communication with the vehicle, unable to send the surveillance data to the vehicle, leading the mission to be *incomplete*

Table 1 illustrates the threat model for the SoS mission. The GPS attack changes the appearance of where the vehicle is located. This leads the vehicle to drive in a different direction, unable to reach the POI, leading the mission to *fail*. It also dislocates the drones because the GPS calculates the distance between the vehicle and the drones. The drone's battery attack is considered to be consuming battery units. The battery consumption attack degrades the drone's capability for surveillance and reconnaissance, which requires more battery. It also reduces the flying time of the drones (e.g., instead of flying 4 h in a regular situation, they will fly 3 h when an attack occurs). If the battery consumption attack occurs, we assume the drones should abort the mission and fly back to the vehicle, and the mission is incomplete. Another possibility is that the drones do not abort, keep flying, lose their battery, and then cannot return to the vehicle. In this case, the mission can fail because the data is not collected, and the vehicle cannot estimate the dangerous situations ahead. If the attacker turns off the camera and infrared sensor, the drones cannot collect data but can fly back to the vehicle, and the mission is incomplete. Besides mission *success*, two mission statuses are defined: mission is *incomplete* and mission *fails*. Mission success indicates that the drones collect data and fly back to the vehicle, and the vehicle reaches the POI and reports to the military base. The incomplete mission indicates that the drones cannot collect data but can fly back to the vehicle, and the vehicle reaches the POI but cannot report to the military base. Mission failure demonstrates that the drones cannot return to the vehicle, the vehicle cannot reach the POI, or the vehicle or the drones do not return to the station. Analyzing the mission status from the mission specification is essential to translate the attacks into formal specifications.

5.2 Formal Specification of Attacks

Each attack described in the threat model is formally specified as a CPN attack (i.e., a sub-CPN model). The CPN attack is plugged into the proper place of the CPN mission based on precondition and postcondition. From the SoS threat model in Table 1, eight CPN attacks are formalized as three CPN attacks related to the vehicle, and five CPN attacks demonstrate the drone attacks. The three CPN attacks of the vehicle are presented in Fig. 4 as vehicle GPS attack, vehicle LOS attack, and BLOS attack. These CPN attacks have the same precondition and postcondition as the *Vehicle* datatype. They must receive a variable of type *Vehicle*, change its values, and return a new state of this variable. These CPN attacks can be attached to any place in the CPN mission if that place has a precondition and postcondition of *Vehicle* datatype.

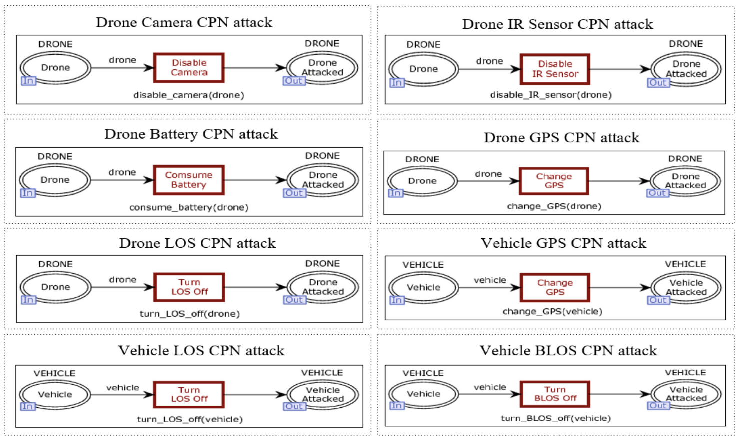

Fig. 4. CPN Attacks for the SoS Mission

Similarly, the five CPN attacks are formalized for the drones: drone GPS attack, battery attack, camera attack, infrared sensor attack, and drone LOS attack. These CPN attacks have precondition and postcondition of datatype *Drone* and must receive a variable of type *Drone*, change its values, and return the new state of this variable. For instance, the CPN battery attack decreases the value of the drone variable's attribute $drone_1.battery$ by 10 units and returns the drone variable with a new state of values. The CPN camera attack changes the value of the drone variable's attribute $drone_1.camera$ to false, and the CPN infrared sensor attack changes the value of the drone variable's attribute $drone_1.IR\_sensor$ to false, and return the drone with new values to the CPN mission. For these changes in the drone's attributes, resiliency analysis is executed to verify where the CPN mission can succeed, be incomplete, or fail.

6 SoS Mission Resiliency Analysis

This section applies Steps 6 and 7 of the methodology to the SoS mission to analyze its resiliency. State space and transitions of the CPN mission attached to CPN attacks are attached are verified using CPNTools [16].

6.1 SoS Mission Formal Specification Analysis

We investigate the decremental resiliency of a mission in which a mission-critical SoS is degraded during execution, and no previously unavailable capability may recover. We start with one attack that degrades a subject's capability. We then decrease subjects' capabilities by different attacks and degrade them to demonstrate a decremental resiliency analysis. Table 2 examines 4 attack scenarios, demonstrating decremental resiliency.

Table 2. Decremental Attack Scenarios

Tasks	Location	Attack	Mission Status
Two drones scan the route and transmit to the military vehicle	Moving to POI	Disable cameras and IR sensors, and consume drones' battery	Incomplete
Two drones scan the route and transmit to the military vehicle	Moving to POI	Change GPS of the drones	Fail
The vehicle and drones are mapping the route and collecting data	Moving to POI	Disable LOS of the drones and the vehicle's BLOS	Fail
Report the situation to the base and wait for engaging instruction	At POI	Disable LOS, cameras and IR sensors of the drones and the vehicle's BLOS	Incomplete

The first attack scenario expresses that the two drones were attacked many times, turning off their cameras and IR sensors and consuming their batteries with 10 units each. These attacks occurred while the SoS was moving toward the POI. The drones could not collect data but had enough battery to safely return to the military vehicle using GPS. The mission status is incomplete because the data was never collected. The second attack scenario shows the drones' GPS being attacked while scanning the route, and they could not maintain the 5-mile distance. They may hover away from the military vehicle and lost. The mission failed because the drones did not return to the military vehicle. The third attack scenario also occurred while the SoS moving toward the POI. The attack disabled the vehicle's BLOS and the drones' LOS, rendering them unable to communicate. The drones could not transmit their GPS and data to the vehicle, nor did the vehicle communicate with the base. The mission failed because the drones could not reach the POI nor return to the vehicle. The fourth attack scenario happened at the POI location, where all communication and sensing capabilities of the vehicle and the drones were disabled. The mission was incomplete because the objectives of collecting data and reporting the base were unmet, but the SoS returned safely to the station.

State Space Analysis. The analysis examines the state space of the CPN model with each attack scenario. Table 3 illustrates the number of nodes and arcs of the state space model (i.e., reachability graph) generated for each attack scenario. It also shows the number of nodes and arcs of the strongly connected components (SCC) graphs generated from the state space models. Table 3 reports that all CPNs of attack scenarios are full SCC. This indicates that the state space models generated from these CPNs are acyclic; they do not have infinite loops. However, it reports dead markings and dead transitions for each attack scenario, which are considered for analysis. Dead markings are states where the CPN is terminated, and dead transitions are those transitions that are not executed (no token visits them). Checking the dead markings gives the values of the SoS attributes when the mission is terminated, whereas analyzing the dead transitions illustrates why the SoS did not execute specific tasks.

Table 3. State Space Report

Location & Attack Scenario	State Space		SCC Graph		Status
Moving to POI, disable cameras and sensors, and consume battery	#Node42	#Arcs 61	#Node42	#Arcs 61	Full
	Dead Markings [30,34,37,42]		#Dead Transition 5		Live Transition None
Moving to POI, change GPS of the drones	#Node31	#Arcs 38	#Node31	#Arcs 38	Full
	Dead Markings [25,26,31]		#Dead Transition 2		Live Transition None
Moving to POI, disable LOS of the drones, and the vehicle's BLOS	#Node37	#Arcs 54	#Node37	#Arcs 54	Full
	Dead Markings [29,33,34,37]		#Dead Transition 5		Live Transition None
At POI, disable drones cameras, sensors, LOS, and vehicle's BLOS	#Node63	#Arcs 85	#Node63	#Arcs 85	Full
	Dead Markings [47,54,55,58,63]		#Dead Transition 7		Live Transition None

We use CPN-ML programming language [8] to verify dead marking and dead transition for each attack scenario. We also use built-in functions provided by the CPNTools [16], such as *Reachable(x, y)*, *AllReachable()*, *NodesInPath(x, y)*, *DeadMarking(x)*, *ListDeadMarkings()*, and *ListDeadTransitions()*, for the verification process. These functions return an execution sequence for each dead marking where the mission is succeeded, incomplete, or failed. We backtrack through the execution sequence and locate the state where the attack occurs. While backtracking, we verify the change of states in the sequence to inspect the reasons that lead to dead transitions and what state values the sequence terminates with at dead markings.

The first attack scenario, moving to POI and disabling cameras and sensors and consuming battery, reports 4 dead markings ([30,34,37,42]) and 5 dead transitions. These dead markings show that the CPN mission terminates where the drone's tokens loop over the battery check transition, which is attacked and ends with an empty mission status. State 42 expresses that the CPN mission terminates with mission status "Incomplete" because the *data_collected* value is false as part of the mission objectives is unsatisfied.

The state space of the second attack scenario, moving to POI and changing the drones' GPS, reports 3 dead markings ([25,26,31]) and 2 dead transitions. States 25 and 26 show that the drones' GPS was attacked in the moving-to-POI location, causing the CPN mission to terminate with an empty mission status. In State 31, the CPN mission completed the mission with mission status "Fail". Verifying this state shows that the drone's GPS differs from the vehicle's GPS, indicating that the drone is not returning to the station. It also shows that the drone *data_collected* attribute is false, and the mission objectives are unmet. The two dead transitions are "$SoS'UseGPStoCheckDistance$ 1" and "$SoS'Checklocation$ 2", indicating that the GPS attack prevented the drones' token from passing through these transitions.

The third attack scenario, moving to POI and disabling the drones' LOS and the vehicle's BLOS, reports 4 dead markings ([29,33,34,37]) and 5 dead transitions. States 29, 33, and 34 show that the CPN mission cannot continue and terminates with an empty mission status. State 37 shows that the CPN mission ended with the mission state "Fail" because the drone's GPS differs from the vehicle's GPS due to disconnection, and the vehicle disconnected from the base. Verifying the 5 dead transitions shows that the vehicle BLOS checks transition leads to check vehicle LOS, uses GPS to check the distance, and checks the location to be dead transitions. It is because these transitions are related to one another; failing one causes failing another. The impact of attacking LOS communication cascades to the GPS capability.

The fourth attack scenario, at POI disabling drones cameras, sensors, LOS, and vehicle's BLOS, reports 5 dead markings ([47,54,55,58,63]) and 7 dead transitions. States 47, 54, 55, and 58 show the CPN mission terminated with an empty mission status. These are because of the number of CPN attacks disabling cameras, sensors, LOS, and BLOS. State 63 shows that the CPN mission ended with the mission state "Incomplete". Even though the number of attacks in the fourth scenario is more than the number of attacks in the third scenario, the CPN mission status ends with "Incomplete" compared to the third scenario that ended with "Fail". In the fourth attack scenario, the drone's GPS is the same as the vehicle's. The drones can collect data because the attacks occurred at the POI location, not the route. The CPN mission status in the fourth attack scenario is incomplete because the SoS cannot report the base; the BLOS is disconnected.

Even though the fourth attack scenario includes more attacks than the third attack scenario, the fourth attack scenario terminates with an incomplete mission status compared to the third attack scenario, which terminates with a failed mission status. Interestingly, the state space of these attack scenarios reports an exact number of dead markings, only one node, and an exact number of dead transitions, 8 dead transitions. The location where the attack scenario occurs significantly varies the mission outcomes.

The verification shows where the SoS mission fails in dispute of attacks. It also points out that attacking one SoS capability affects others, and the mission fails. Lastly, the impact of attacks is different from one location to another (geographically); an attack is critical at one location and ineffective at another. SOS engineering must take into account these facts when describing resilient SoS missions.

6.2 SoS Mission Resiliency Analysis

The verification indicates that the SoS must immediately abort the mission in some instances, especially when LOS and BLOS communication is attacked. Continuous surveilling and moving toward the POI without communication minimizes the resilience level of the mission. To enhance the resiliency of SoS critical missions, suitable restrictions must be counted on the SOS missions, and improvements for SoS capabilities must be considered. Table 4 expresses constraints and improvements to increase the resiliency level of the SoS critical-missions. That is, if we add these constraints and improvements to the SoS mission described in Sect. 2, this mission becomes resilient and resistant to various types of attacks.

Table 4. Mission Restrictions & Improvements

Task	Location	Restriction & Improvement
Drones scan the route and collect data	Moving to POI	Abort the mission if the drones' LOS are disabled. Provide recoverable LOS to pursue the mission
Drones scan the route and collect data	Moving to POI	Abort the mission if the drones' LOS are attacked. Or, provide recoverable LOS to pursue the mission
Drones scan the route and collect data	Moving to POI	Provide recoverable cameras and IR sensors for the drones as well as backup battery
Military vehicle moves toward POI	Moving to POI	Abort the mission if the vehicle' BLOS is attacked. Or, provide recoverable BLOS to communicate with the base
SoS reports POI situation to the base	At POI	Use backup communication channels when BLOS is attacked and provide recoverable BLOS for reporting

7 Related Work

The literature on resiliency issues discusses the unavailability [12–14, 18, 20]. We propose that the resiliency problem can be considered as degradation.

Wang *et al.* [18] introduce three types of resilience, static, decremental, and dynamic. They use the term a user to refer to a subject, an active entity doing tasks. Static resilience refers to a situation in which subjects become unavailable before the workflow executes, and no subjects may become available during the execution. Decremental resiliency expresses a condition where subjects become unavailable before or during the execution of the workflow, and no previously unavailable subjects may become available during execution, while dynamic resilience describes the situation where a subject may become unavailable at any time; a previously unavailable subject may become available at any time. The different types of resilience formulations capture various types of attack scenarios.

Yang *et al.* [20] address the workflow satisfiability problem that ascertains whether a set of subjects can complete a workflow. The authors investigate the computational complexity of the workflow satisfiability problem in two aspects. One aspect considers either one path or all paths of a workflow, and the other focuses on the possible patterns in a workflow. They present a set of algorithms for solving various types of problems of workflow satisfiability. They show that many existential and universal workflow satisfiability problems are NP-complete and NP-hard. Thus, they conclude that restrictions on workflow patterns induce such problems to be solvable in polynomial time.

Mace *et al.* [12–14] propose a quantitative measure of workflow resiliency. They use a Markov Decision Process (MDP) to model workflow to provide a quantitative measure of resilience. They refer to binary classification, such as returning an execution sequence if one exists and declaring the workflow resilient or returning false and declaring the workflow not resilient. The authors show that the MDP models give a termination rate and an expected termination step.

8 Conclusion

This paper introduces a methodology that addresses the mission resiliency analysis for mission-critical System of Systems (SoS). The methodology systematically analyzes various decremental resiliency of SoS missions. The methodology uses workflow for the formal representation of the SoS mission. It also constructs threat models and converts them into formal representation of attacks. The methodology then utilizes Coloured Petri Nets (CPN) to specify the SoS missions with attacks attached.

We applied the methodology to a SoS formation demonstrating a military vehicle and two drones working together to monitor a national border. The results show that the resiliency of the SoS mission is a degradation issue. When attacks occur, a mission-critical SoS cannot continue, and the mission fails or is incomplete. The methodology indicates which state the SoS mission succeeds, fails, and is incomplete. It also highlights restrictions that must be added to the SoS mission to improve resiliency.

Future work will investigate the methodology's application to dynamic resiliency, where a system is able to recover its components after an attack. The model state explosion will be investigated to verify such scenarios. Future work will also use threat modeling frameworks such as STRIDE and PASTA to cover various types of attacks that violate the confidentiality, integrity, and availability in the context of a mission.

Acknowledgements. This work was supported in part by funding from NIST under Award Number 60NANB23D152 and NSF under Award Numbers CNS 2335687, DMS 2123761, CNS 1822118, NIST, ARL, Statnett, AMI, NewPush, and Cyber Risk Research.

References

1. Abdelgawad, M., Ray, I., Vasquez, T.: Workflow Resilience for Mission Critical Systems. In: Dolev, S., Schieber, B. (eds.) Stabilization, Safety, and Security of Distributed Systems. SSS 2023. LNCS, vol. 14310. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-44274-2_37
2. Cook, S., Pratt, J.: Advances in systems of systems engineering foundations and methodologies. Aust. J. Multi-Disciplinary Eng. **17**(1), 9–22 (2021)
3. CPN Tools Administration: Help topics for verification (2018). http://cpntools.org/2018/01/15/verification/. Accessed 12 Jan 2024
4. Dahmann, J., Roedler, G.: Systems of systems engineering standards. Insight **19**(3), 23–26 (2016)
5. Figueira, N., Pochmann, P., Oliveira, A., de Freitas, E.P.: A C4ISR application on the swarm drones context in a low infrastructure scenario. In: the proceedings of the International Conference on Electrical, Computer and Energy Technologies (ICECET), pp. 1–7. IEEE, Prague, Czech Republic (2022)
6. Houliotis, K., Oikonomidis, P., Charchalakis, P., Stipidis, E.: Mission-critical systems design framework. Adv. Sci. Technol. Eng. Syst. J. **3**(2), 128–137 (2018)
7. ISO/IEC/IEEE: International standard – systems and software engineering – system of systems (SoS) considerations in life cycle stages of a system. ISO/IEC/IEEE 21839:2019(E) **1**(8767114), 1–40 (2019)
8. Jensen, K., Kristensen, L.M.: CPN ML programming. In: Coloured Petri Nets, pp. 43–77. Springer, Heidelberg (2009). https://doi.org/10.1007/b95112_3
9. Jensen, K., Kristensen, L.M.: Coloured Petri Nets. Springer, Heidelberg (2009). https://doi.org/10.1007/b95112
10. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri Nets and CPN tools for modelling and validation of concurrent systems. Int. J. Softw. Tools Technol. Transfer **9**(3–4), 213–254 (2007)
11. Lappas, V., et al.: Autonomous unmanned heterogeneous vehicles for persistent monitoring. Drones **6**(4), 94 (2022)
12. Mace, J.C., Morisset, C., van Moorsel, A.: Quantitative workflow resiliency. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 344–361. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_20
13. Mace, J., Morisset, C., van Moorsel, A.: Modelling user availability in workflow resiliency analysis. In: the proceedings of the 2015 Symposium and Bootcamp on the Science of Security (HotSoS), pp. 1–10. ACM, Urbana Illinois, USA (2015)
14. Mace, J.C., Morisset, C., van Moorsel, A.: Resiliency variance in workflows with choice. In: Fantechi, A., Pelliccione, P. (eds.) SERENE 2015. LNCS, vol. 9274, pp. 128–143. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23129-7_10
15. Ponsard, C., Massonet, P., Molderez, J.F., Rifaut, A., Lamsweerde, A.V., Van, H.T.: Early verification and validation of mission critical systems. Formal Methods Syst. Des. **30**(3), 233–247 (2007)
16. Ratzer, A.V., et al.: CPN tools for editing, simulating, and analysing coloured petri nets. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 450–462. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44919-1_28
17. Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., McQuaid, R.: NIST special publication 800–160 volume 2: developing cyber resilient systems. NIST Spec. Publ. **2**, 224 (2019)
18. Wang, Q., Li, N.: Satisfiability and resiliency in workflow authorization systems. ACM Trans. Inf. Syst. Secur. **13**(4), 1–35 (2010)

19. Weisman, M., et al.: Quantitative measurement of cyber resilience: Modeling and experimentation. Computing Research Repository (2023). (CoRR) ABS/2303.16307
20. Yang, P., Xie, X., Ray, I., Lu, S.: Satisfiability analysis of workflows with control-flow patterns and authorization constraints. IEEE Trans. Serv. Comput. **7**(2), 237–251 (2014)