Resiliency Graphs: Modelling the Interplay between Cyber Attacks and System Failures through AI Planning

Shadaab Kawnain Bashir, Rakesh Podder, Sarath Sreedharan, Indrakshi Ray, and Indrajit Ray
Computer Science Department, Colorado State University, Fort Collins, Colorado, USA
{Shadaab.Bashir, Rakesh.Podder, Sarath.Sreedharan, Indrakshi.Ray, Indrajit.Ray}@colostate.edu

***Abstract*—Operation efficiency in cyber physical system (CPS) has been significantly improved by digitalization of industrial control systems (ICS). However, digitalization exposes ICS to cyber attacks. Of particular concern are cyber attacks that trigger ICS failure. To determine how cyber attacks can trigger failures and thereby improve the resiliency posture of CPS, this study presents the Resiliency Graph (RG) framework that integrates Attack Graphs (AG) and Fault Trees (FT). RG uses AI planning to establish associations between vulnerabilities and system failures thereby enabling operators to evaluate and manage system resiliency. Our deterministic approach represents both system failures and cyber attacks as a structured set of prerequisites and outcomes using a novel AI planning language. AI planning is then used to chain together the causes and the consequences. Empirical evaluations on various ICS network configurations validate the framework's effectiveness in capturing how cyber attacks trigger failures and the framework's scalability.**

***Index Terms*—resiliency graph, cyber physical system, attack graph, fault tree, AI planning.**

I. Introduction

Cyber Physical System (CPS) operations have witnessed a radical transformation in the last decade through technological developments in the Operational Technology (OT) of Industrial Control Systems (ICS), in particular digitalization. While it brings numerous benefits and improvements to ICS, digitalization also exposes the ICS OT to new and sophisticated cyber-attacks. These cyber-attacks can in turn trigger ICS failures in manners not previously thought of. When software vulnerabilities in Information Technology (IT) are combined with system failures of the Operational Technology (OT), attackers get access to new attack surfaces. For example, a software vulnerability may not be instantly exploitable in normal working conditions, but in the event of a network failure, it can become a critical point of exploitation. Such situations can trigger safety events in the CPS causing devastating outcomes.

IT practitioners have often used *Attack Graphs* (AG) to analyze their systems for security vulnerabilities and leverage that analysis to deploy a robust cybersecurity defense. Similarly, ICS operators have used *Fault Trees* (FT) to prepare for safety issues in ICS. AG can identify likely paths an attacker could take to exploit system vulnerabilities, with a focus on cyber threats, while FT illustrates the logical relationships between system failures and their root causes. However, neither AG nor FT can by themselves identify how cyber attacks can trigger safety failures in CPS. Moreover, to our knowledge, there is no such framework that can help analyze how a system vulnerability can trigger a safety event, resulting in cascading failures in a CPS.

Towards this end, we present a new framework which we call *Resiliency Graph* (RG), to improve the security posture of CPS. This framework is developed from a composition of AG depicting cyber-attacks in the CPS, and FT indicating failures. RG uses AI Planning to identify the relationship between vulnerabilities with the corresponding faults that the vulnerability can trigger. This framework will allow the CPS operators to analyze, evaluate, and manage the safety profile of their systems. We particularly use AI planning to develop our framework since it can be scaled to very large problem spaces which is important for a CPS.

Thomas et al. [1] patented a methodology for assessing the impact of cyberattacks on CPS. However, they used a probabilistic approach to map undesirable changes in CPS behavior caused by cyberattacks. As opposed to this, our technique is deterministic and is based on a formal representation of the pre/post-conditions of both attacks and failures. This method prevents the ambiguities that probabilistic approaches introduce and guarantees the accuracy and mathematical validity of the solution provided.

In this paper, we provide a planning formulation that adequately represents cascading failure analysis. Although there have been previous studies [2]–[5] on learning planning representations for AG, these methods are not applicable here since the representation must be supplemented by information from various sources, such as physical failures. To address this, we offer an algorithm that extends an RG populated with attack information by integrating failure data obtained from interactions with a physical system. This integration enables a more thorough and accurate assessment of system resilience in the face of cyber and physical threats.

In this work, we make the following major contributions:

1) We present a novel modeling paradigm, *Resiliency Graph* (RG) that combines the power of Natural Language Processing (NLP) and AI Planning (PDDL) to evaluate and analyze how a cyber-attack can trigger safety events in the CPS.

2) We developed an algorithm called *Resilience Path Learning Algorithm* (RPLA) which autonomously models the dependencies between vulnerabilities and system faults within a resiliency graph to enhance ICS resiliency analysis.
3) Using the *Resiliency Graph* (RG), we have demonstrated how the cascading effects of safety events can be captured, analyzed, and evaluated.
4) A series of empirical assessments on networks with various configurations are conducted to prove the scalability and applicability of our approach.

The paper is structured as follows: Section II reviews relevant literature and highlights major findings and limitations from previous studies. The background on AI Planning is discussed in Section III. In Section IV, we formally describe our new modeling paradigm RG. The steps used to construct RG are described in Section V. In Section VI, we perform analysis on cascading failure. An empirical evaluation of RG is conducted in Section VII. In the end, Section VIII presents the importance of the research issue and how this new framework will contribute to the field.

II. Related Work

Network vulnerabilities are often comprehended and managed using attack tree/graphs analysis (see, for instance, [6]–[13]). This approach has its roots in FT analysis [14]–[17]. The ICS community has been constantly looking for potential issues resulting from safety events due to component failures and accidental human errors.

Since the introduction of attack graphs by Phillips et al. in 1998 [18], the field has seen significant advancements. Shahriari and Jalili developed a polynomially bounded algorithm [19] for analyzing network vulnerabilities using graph-based techniques. Sheyner et al. introduced tools for automated attack graph creation and analysis [20], leveraging the NuSMV [21] model-checker to evaluate security measures. Hankin et al. introduced "Attack Dynamics" [22], a tool using the CAPEC database for detailed attack path visualization. Jajodia et al. enhanced this with their Topological Vulnerability Analysis [23], which examines dependency among vulnerabilities. Kerem et al. proposed a distributed search method in a multi-agent setup [9], addressing large network management and state space issues. Ingols et al. developed NetSPA [24], which uses MP graphs for swift and scalable network evaluations, highlighting areas for further improvement in recommendation algorithms and graph visualization.

Not only in AG generation but using AG as a tool to analyze security incidents, researchers showed the importance of AGs in cybersecurity. Lee et al. introduced ontology-based semantic attack graphs [25], enhancing machine readability and addressing scalability. Noel and Jajodia simplified these graphs by consolidating attack paths into broader nodes [26], reducing complexity for analysts. Hong and Kim proposed a dual-layer graph architecture [27], which separates host vulnerabilities and network topology, enhancing distributed processing and reducing overhead. Mehta et al. adapted Google's PageRank to assess node reachability in AG [28], providing a probabilistic analysis. Templeton and Levitt's model [29] aids administrators in identifying vulnerabilities and formulating countermeasures. Ibrahim et al. also employed AADL in CPS [30], visualizing attack sequences to support defense strategies. Wang et al. integrated attack graphs with a hidden Markov model [31] for analyzing security state probabilities and introduced a breadth-first search algorithm [32] for security metrics based on attack likelihood. Kordy et al.'s ADTool [33], based on Attack-Defense Trees, facilitates modeling and real-time analysis of security scenarios. Lallie et al. compared the Adapted Attack Graph and Fault Tree methods [34], focusing on enhancing cyber-attack comprehension across user groups.

However, this work faces an inherent challenge, scalability. To tackle this issue, Ammann et al. used the concept of monotonicity to develop a scalable graph-based model [35], using a Java-based algorithm to identify minimal attack chains without exhaustive search to encode attack trees. Ou et al. [8] also proposed a framework that acknowledges the challenges of scalability. Khouzani et al. [36] provide a mathematical framework for cyber-security planning in which the problem is formulated as a multi-objective bi-level optimization, with security risks and control costs balanced. Their approach is scalable and capable of handling large AGs.

Similarly, in the area of fault analysis, there have been many notable works. Event Tree Analysis (ETA) is a technique used in probabilistic risk assessment, often applied in high-stakes industries like nuclear power, to understand potential accident sequences following a specific initiating event or set of events [37]. Various techniques have been employed to construct an ETA, for instance, Andrews et al. developed a technique using binary decision diagrams (BBD) [38] to effectively manage complex event trees and streamline probability computations. In a distinct approach, Kenarangui employed the principles of fuzzy logic [39] to ETA, resulting in a system where a range of 'fuzzy' probabilities are assigned instead of specific values to better handle uncertainty.

Fault Tree Analysis (FTA) is a deductive, top-down method used for system reliability and safety studies. Originally developed at Bell Telephone Laboratories in 1962 for the analysis of the intercontinental Minuteman missile's launch control system [14], it is now widely used in industries such as nuclear power plants, aerospace, and defense. Modern FTA includes various types, from static, and dynamic to non-coherent fault trees, and employs classical and modern techniques, with advanced topics covering importance analysis, common-cause failures, and application of fault trees in analyzing multi-state systems and phased-mission systems.

Failure Mode and Effects Analysis (FMEA), initially introduced in the aerospace industry [40], is a method used to predict and prevent failures in systems, processes, or services. This method aims to identify potential failure modes, understand their causes and effects, and consequently formulate strategies to mitigate the likelihood of these failures [41].

Crucitti et al. [42] presented a model that depicts cascading failures in convoluted networks. Using the concept of dynamic

redistribution of flow within a network they proved how the failure of the node with the highest load might result in the decline of the network efficiency. However, in their work they mainly employed specific network topologies such as Erdős-Rényi (ER) and Barabási-Albert (BA) networks, limiting their findings to other types of networks. Wang et al. [43] develop a method for identifying vulnerabilities in the Industrial Internet of Things (IIoT), which improves attack path quantification and streamlines path-finding. This method involves creating a vulnerability graph model from AG using a maximum loss flow algorithm. Their approach combines a cyber simulation model with a control simulation model to detect undesirable changes in CPS behavior caused by errors or failures. This mapping procedure identifies which cyber events can cause these defects or errors by examining the lexical similarities of block names in the cyber and control simulation models, as well as the structural similarities between their components. However, due to the probabilistic nature of this mapping, users must choose probabilistic thresholds. This can lead to mistakes in the analysis, such as false positives (identifying a safe condition as harmful) and false negatives (failing to identify a true threat). Furthermore, it is unclear how these probabilities might be used to estimate the likelihood of cascading failures, if at all, in which one failure causes a chain of subsequent failures. In contrast, our technique is deterministic, which is based on a formal representation of the pre-conditions and post-conditions of both attacks and failures. This formalism ensures that the solution is accurate and mathematically valid, avoiding ambiguities that probabilistic approaches introduce.

However, utilizing AG for cyber resiliency analysis in OT networks, especially analyzing for the cascading failure triggered by cyber attacks, presents several challenges. Unlike FT, AG lacks a standardized formalism. Various researchers model the concept of an AG differently, leading to variations in their properties. There is a research gap between AG and FT [44]. It is quite evident that cyber attacks and safety issues are interlinked. For, example an attack can cause a fault in the OT system, similarly, a fault can create a vulnerability that an attacker uses to exploit further. Thomas et al. [1] patented a methodology for assessing the impact of cyber attacks on CPS, which is comparable to the approach we are proposing. In their framework, they used a cyber and control simulation model to detect behavioral changes caused in CPS due to safety failures by cyber attacks. This probabilistic nature of mapping can lead to false positives and false negative errors. It is also unclear how their framework estimates the likelihood of cascading failures. Our approach, on the other hand, avoids ambiguities ensuring correctness and mathematical validity for its deterministic nature. From the above study, it is clear that there are many works on AG and FTA but to our knowledge, there is no formal framework that can also represent undesirable fault and failure states in OT networks caused by vulnerabilities.

III. Background

A standard planning problem is represented by a tuple of the form $\mathcal{M} = \langle F, A, I, G \rangle$, where F is a set of propositional fluents or variables that defines the state space of the model. Each state s is uniquely defined by the set of fluents or variables that are true in that state $s \subseteq F$. A denotes the set of executable actions, I represents the initial state ($I \subseteq F$), and G specifies the goal description ($G \subseteq F$). A planning problem can be represented as a directed graph, known as a transition system. In this graph, the nodes represent the different states of the system. The directed edges in the graph indicate transitions between states that occur as a result of executing an action. Each action $a \in A$ is further specified by the tuple $a = \langle \mathit{pre}(a), \mathit{add}(a), \mathit{del}(a) \rangle$. Here, $\mathit{pre}(a) \subseteq F$ represents the preconditions that determine the states in which the action can be executed. The set $\mathit{add}(a)$ includes the add effects, specifying the fluents that will be set to true upon executing the action, while $\mathit{del}(a) \subseteq F$ comprises the delete effects, indicating the fluents that will be set to false upon execution. The outcome of executing an action in a state s is captured by a transition function $\Theta_{\mathcal{M}}$, such that

$$\Theta_{\mathcal{M}}(s, a) = (s \setminus \mathit{del}(a)) \cup \mathit{add}(a), \text{if } \mathit{pre}(a) \subseteq s. \quad (1)$$

The planning problem involves identifying a path from the initial state to a goal state. A solution to this problem, known as a plan, is represented by a sequence of actions. An action sequence $\pi = \langle a_1, ..., a_k \rangle$ is considered a valid plan if $\Theta_M(I, \pi) \supseteq G$.

Ghosh [45] used a variant of Planner called SGPlan to generate a minimal attack graph in polynomial time which makes the attack graph scalable and avoids the state-space explosion problem. Yichao et al. [46] proposed a compact graph planning-based attack paths discovery algorithm using a depth-first backward search algorithm to discover hidden attack paths. There are many other works [47]–[49] that use variations of classical planning to discover attack paths. Previous research has not investigated the impact of cyber attacks on CPS. Therefore, in this paper we introduces the *Resiliency Graph* (RG) framework, which elucidates how cyber attacks can precipitate safety incidents within CPS.

IV. Resiliency Graph

To maintain system resiliency, it is critical to understand the relationship between attacks and the faults caused by them in the system. This allows the organization to continue operating and recovering regardless of adversarial attacks and random faults. To capture this dynamic interactions, we introduce the concept of a new modeling paradigm called *Resiliency Graph* (RG). For a given network, RG consists of a group of hosts or computers organized in a particular network topology each identified by a label uniquely. An attribute set consisting of propositional facts is used to capture the host-level information of the network such as host configurations, vulnerabilities, faults, etc, and are represented as true and false values for a given propositional fact.

***Definition** 1: The attribute set C_H is a set of propositions that could be true for a host in a given network. Such as $C_H = \{c_1, c_2, \ldots, c_n\}$, where c_i is a proposition about a host's attribute. We denote the set of unique hosts and their*

labels using H, $H = \{H_1, H_2, \ldots, H_n\}$. Each host $H_i \in H$ has a specific set of attributes from C_H that are true for that host.

The set of pre-conditions for a specific attack/fault is represented as pre. Hence, for an attack/fault to be feasible on host H_i, $pre^i \subseteq C_i$ where C_i is the set of attributes true for host H_i. Additionally, $post = \langle post^+, post^- \rangle$ represents the set of post-conditions for a specific attack/fault. After an attack/fault on host H_i, the positive post-conditions $post^{i+}$ will be added to the set of attributes, and the negative post-conditions $post^{i-}$ will be removed from the set of attributes.

RG comprises of hosts/nodes, attack edges, and fault edges, illustrating the paths through which attacks and faults propagate. The RG enables the analysis of paths that combine attack and fault sequences, ultimately leading to a resilient solution from a source to a destination node.

***Definition** 2: A **Resiliency Graph** RG is defined as a directed graph $RG = \langle \mathcal{N}, \mathcal{A}, \mathcal{F} \rangle$ where $\mathcal{N}$ is the set of nodes and are defined as $\mathcal{N} = H \times 2^{C_H}$, each representing a host in the network. $\mathcal{A} = \{(n_i, n_j) \mid n_i, n_j \in \mathcal{N}\}$ is the set of directed edges called attack edges, where each edge (n_i, n_j) represents a potential attack. $\mathcal{F} = \{(n_i, n_j) \mid n_i, n_j \in \mathcal{N}\}$ is the set of directed edges called fault edges, where each edge (n_i, n_j) represents a potential fault path from node n_i to node n_j. For each possible attack and fault edge, $e \in \mathcal{A} \cup \mathcal{F}$, the pre-conditions ($pre$) and post-conditions $\langle post^+, post^- \rangle$ have to be true, where*

- *$pre^i(e) \subseteq C_i$, where $n_i = (H_i, C_i)$*
- *for the node $n_j = (H_j, C_j)$, we have $C_j \subseteq post^{+j}(e)$ and $C_j \cap post^{-j}(e) = \emptyset$*

Additionally, let $S_0 = (n \in \mathcal{N}) = \{n_i\}$ be the source node and $D_t = (n \in \mathcal{N}) = \{n_j\}$ be the destination node. The current graph takes the form of an OR graph, however, it can be easily extended to support resiliency graphs that need to be represented as AND-OR graphs. The PDDL formalism that we are using to concisely capture RG graphs, can also be used to capture AND-OR graphs easily.

***Definition** 3: An **attack path** is a path through the graph between two arbitrary nodes and it only contains attacks.*

***Definition** 4: A **fault path** is a path through the graph between two arbitrary nodes and it only contains faults.*

***Definition** 5: A **resilience path** P_R from S_0 to D_t is a sequence of nodes $(S_0, n_1, \ldots, n_k, n_{k+1}, \ldots, D_t)$ where there exists a partitioning of the path such that:*

- *$(n_i, n_{i+1}) \in \mathcal{A} \cup \mathcal{F}$ for $i \in \{0, 1, \ldots, t\}$.*
- *There exists at least one sub-path composed of attack edges followed by a sub-path composed of fault edges, or vice versa.*

An attack is initiated at the source node S and traverses through a sequence of attack edges $(n_i, n_{i+1}) \in \mathcal{A}$ until reaching an intermediate node n_k. Upon compromising node n_k, a fault/failure is triggered, activating the traversal of fault edges $(n_i, n_{i+1}) \in \mathcal{F}$. Continue the traversal through a sequence of fault edges from n_k to the destination node D. The resilience path P_R is constructed by combining both the attack and fault phases, ensuring a continuous path from S to D.

***Definition** 6: For a given RG, initial node state S, and goal state D has attribute C_d, a representative planning model is defined as $\mathcal{M} = \langle F, A, I, G \rangle$, where*

- *F contains a fluent for each host attribute pair in $H \times C_H$, where we use the function $\delta^{\mathcal{M}}$ to capture the mapping from $H \times C_H$ to F.*
- *A contains an action for each attack or fault edge in $e \in \mathcal{A} \cup \mathcal{F}$. Let a_e be the action corresponding to an edge e from node $n_i = (H_i, C_i)$ to node $n_j = (H_j, C_j)$. Here the action precondition is given as pre$(a_e) = \{\delta^{\mathcal{M}}(H_i \times pre)\}$. It's an resiliency edge, $\langle pre, post^+, post^- \rangle$, then add$(a_e) = \delta^{\mathcal{M}}(\{H_i\} \times post^+)$, and del$(a_e) = \delta^{\mathcal{M}}(\{H_i\} \times post^-)$.*
- *$I = \{\delta^{\mathcal{M}}((H_s, c_s)\}$, and $G = \{\delta^{\mathcal{M}}((H_t, c_t)\}$.*

With this mapping, our model of RG can be represented accurately without loss of information.

V. Methodology

In this section, we discuss the steps for learning the *Resiliency Graph* (RG) and identifying the *Resilience Path* (P_R).

A. Resilience Path Learning Algorithm

To generate the RG, which is a combination of a number of P_R, we develop an algorithm named the *Resilience Path Learning Algorithm* (RPLA) described in Algo. 1. It connects the vulnerabilities with the safety events they trigger using AI planning. Initially, we start with an optimistic representation of an underlying model in PDDL, which we iteratively refine towards the true representation is expressed as, $\mathcal{M}^{\mathcal{E}} = \langle F^{\mathcal{E}}, A^{\mathcal{E}}, I^{\mathcal{E}}, G^{\mathcal{E}} \rangle$, where $F^{\mathcal{E}}$ is a set of fluents, $A^{\mathcal{E}}$ represents set of actions, $I^{\mathcal{E}}$ is the initial state and $G^{\mathcal{E}}$ denotes the goal state, has been created. We also assume that we have a base model $\mathcal{M}^{\mathcal{S}}$ that contains the ground truth information of the entire system necessary to build an RG. This base model, in our case, acts as a black box and only provides the agent information about the state when an action $a \in A$ is performed. The agent can also reset the base model's current state after each iteration.

At the beginning of the Algo. 1, the initial states of both the $\mathcal{M}^{\mathcal{S}}$ and $\mathcal{M}^{\mathcal{E}}$ are stored in s^S and $s^{\mathcal{E}}$ respectively. Afterward, a plan is generated for the model $\mathcal{M}^{\mathcal{E}}$ using A* search. Next, in order to produce the P_R, the algorithm begins updating the post-conditions of each action in the plan of the $\mathcal{M}^{\mathcal{E}}$ iteratively. Initially, it pops an action a from the plan and checks whether the pre-conditions of the a are the subset of the current state of the $\mathcal{M}^{\mathcal{S}}$. If the condition is true, it updates the current state of the $\mathcal{M}^{\mathcal{E}}$ by adding the post-conditions of a, and concurrently it finds the state of the $\mathcal{M}^{\mathcal{S}}$ after executing action a. The algorithm also looks for dissimilarities between the current state of the $\mathcal{M}^{\mathcal{S}}$ and $\mathcal{M}^{\mathcal{E}}$ after executing action a. Then it removes the extra fluents, that are not present in the states of $\mathcal{M}^{\mathcal{S}}$, from the current states and the post-conditions

Algorithm 1: Resilience Path Learning Algorithm

Input : $\mathcal{M}^{\mathcal{E}} = \langle F^{\mathcal{E}}, A^{\mathcal{E}}, I^{\mathcal{E}}, G^{\mathcal{E}} \rangle$
Output: plan

$s^{S} \leftarrow$ INITIALSTATE($\mathcal{M}^{\mathcal{S}}$)
$s \leftarrow I^{\mathcal{E}}$
$plan, x \leftarrow$ GENERATEPLAN($\mathcal{M}^{\mathcal{E}}$)
if *plan* $\neq \emptyset$ **then**
  while $x \neq \emptyset$ **do**
    $a \leftarrow x.pop()$
    if $pre(a) \subseteq s^{S}$ **then**
      $s^{S} \leftarrow$ CURRENTSIMSTATE(a)
      $s^{\mathcal{E}} \leftarrow s \cup post(a)$
      if $s^{S} \neq s^{\mathcal{E}}$ **then**
        $f =$ EXTRAFLUENTS($s^{S}, s^{\mathcal{E}}$)
        remove f *from* $add(a)$
        remove f *from* $s^{\mathcal{E}}$
      end
      if $G \subseteq s^{\mathcal{E}}$ **then**
        return $plan$
      end
    end
  end
else
  No plan exist
end

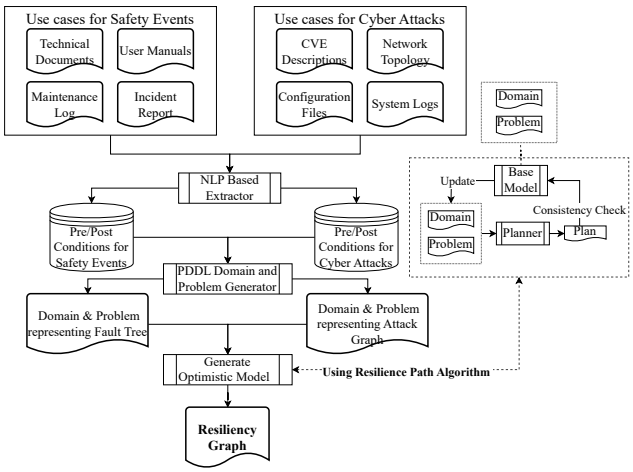


Fig. 1: Workflow representing steps involved in generating RG from Attack Graphs and Fault Trees of CPS.

of action a of $\mathcal{M}^{\mathcal{E}}$. This step ensures the consistency and alignment between the $\mathcal{M}^{\mathcal{S}}$ and $\mathcal{M}^{\mathcal{E}}$ states. At each step of this iterative process, the algorithm checks whether the goal G is a subset of the $\mathcal{M}^{\mathcal{E}}$ current state, and once the goal is reached the P_R has been identified. It returns the plan that leads to this P_R. Finally, for various initial and goal states our algorithm generates an RG through combining multiple P_R for the entire system by updating the actions required to reach the particular goal.

The limitation of RPLA is that it can not create all P_R, that exists in the system at once. It can only create the P_R based on the selected initial and goal state of the optimistic model.

***Proposition** 1: The Algorithm 1 will always identify a Resilience Path's plan if one exists, else it will return an empty plan if there is no P_R exists.*

The model is an optimistic approximation [50], which means the model only allows for a super set of true plans, so if plan is not found, it can not exist.

B. Workflow Overview

The overall architectures for developing the RG framework for CPS is illustrated in Fig. 1. This novel approach unifies security-related data usually represented by AGs and safety-related data usually modeled with FTs. At first, the data related to safety events and cyber attacks are collected from user manuals, technical reports, CVE descriptions, network topology, etc. Then, these data are given to an NLP-based extractor which analyzes the textual data, and extracts all the necessary pre/post conditions for safety events and cyber-attacks. Pre-conditions refer to the states that need to hold for an event to take place while post-conditions are the states that will hold following the occurrence of the event. After extracting the pre/post-conditions, it is forwarded to the PDDL generator to generate specific domain and problem files for AG using AGBuilder [4] and FT by domain-expert. The base model will provide ground truth about the state when an action is performed. The optimistic model's state space will change by using Algo. 1 by iteratively updating the post-conditions of the actions based on the base model's output. Thus the generation of a comprehensive RG framework that takes into account the system's security and safety postures for the CPS.

VI. Resiliency Graph Analyses

To illustrate the workflow of *Resiliency Graph* (RG) we are using Flare System as a testbed. We also present a precise and systematic representation of RG for this system. Finally, a detailed analysis of how RG is able to capture the cascading failure is discussed.

A. System Description

We are using LNG complex at GL1/Z-SONATRACH - Algeria [51], [52], which comprised three flare systems: a cold flare system for gases of a temperature lower than 0°C, a hot flare for gases of a temperature higher than 0°C, and the tank flare system for excess vapors from the LNG storage tanks as test-bed.

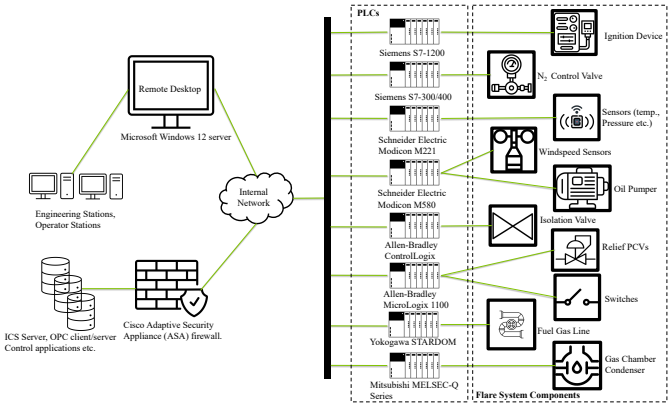


Fig. 2: Overview of OT and IT component of Flare System.

PLC Name	#CVE Name	Operation in OT Network
Siemens S7-1200	CVE-2017-12741, CVE-2018-13810	Control the Ignition Device.
Siemens S7-300/400	CVE-2017-12741, CVE-2016-8673	Control the opening/closing of Nitrogen Valve.
Schneider Electric Modicon M221	CVE-2017-6032, CVE-2017-6034	Monitor the Mechanical & Instrumental reading of sensors.
Schneider Electric Modicon M580	CVE-2019-6815, CVE-2018-7789	Monitor the windspeed and pumping of flare system.
Allen-Bradley ControlLogix	CVE-2017-9312	Control the isolation valve.
Allen-Bradley MicroLogix 1100	CVE-2017-12089	Control the relief PCVs and switching of flares.
Yokogawa STARDOM	CVE-2018-10594	Control and monitor Fuel Gas line.
Mitsubishi MELSEC-Q Series	CVE-2017-9638	Monitor and control the condensate presence in Fuel Gas.

TABLE I: List of PLCs, their associated CVE vulnerabilities and functions in Flare System.

The Flare System is an OT system, which uses hardware and software to monitor and control physical processes, devices, and infrastructure. In the Flare System, various Programmable Logic Controllers (PLCs) are responsible for automating and controlling various processes to ensure that the flare operates safely as illustrated in Fig. 2, efficiently, and in compliance with environmental regulations. Table I shows all the PLCs and which physical aspects of the Flare System it is controlling or monitoring, and a list of vulnerabilities associated with each PLC. These PLCs are connected to the remote desktop with a Microsoft Windows 12 server and a Cisco Adaptive Security Appliance (ASA) firewall. An adversary can access the internal network by exploiting various vulnerabilities (CVE-2019-0575, CVE-2019-0584, CVE-2018-0538) and executing a remote code or exploiting the vulnerabilities (CVE-2018-0296) of the firewall.

B. RG Construction of Flare System

To construct RG utilizing the *Attack Graph* (AG) and *Fault Tree* (FT) shown in Fig. 14 (appendix), we first build a base model to represent the Flare System. For the purpose of this experiment, this base model is constructed using PDDL, a well-established technique to convert attack graphs and associated faults to a planning problem. The process of generating the base model using PDDL for some parts is dependent on domain experts [44]. Also, with each addition of new systems the base model is subjected to an incremental refinement to address new security concerns and safety issues.

Initially, we model an AG (as domain and problem file) for the Flare System are shown in Fig. 3 and Fig. 5 respectively capturing the pre and post-conditions of cyber attack in that system constructed using AGBuilder. Similarly Fig. 4 and Fig. 6 represent the (FT), capturing the pre and post-conditions of safety events in that system constructed using the help of domain experts using PDDL. Fig. 7, illustrates an optimistic representation of the Flare System where each action's post-conditions contain all the possible faults that can be triggered within the system without knowing which vulnerabilities are responsible for triggering a safety event. Fig. 8 represents the problem file for the Flare System which contains the initial

```
(define (domain flare-attack-graph-domain)
(:requirements :strips :typing)
(:types: node)
(:predicates
    (has-compromised-customer-pc ?node - node)
    (has-vulnerability-CVE-2017-6032 ?node - node)
    (plc-compromised ?node - node) ...)
(:action access-to-windows-server
    :parameters (?at - node)
    :precondition (and
        (has-compromised-customer-pc)
        (has-compromised-engineering-workstations))
        :effect (and
        (has-access-to-windows-server ?at)))
.
.
.
(:action exploits-vulnerability-CVE-2017-6032
    :parameters (?at ?to - node)
    :precondition (and
        (has-connected ?at ?to))))
        (has-access-to-network ?at)
        (has-vulnerability-CVE-2017-6032 ?to))
    :effect (and
        (has-Schneider-Electric-Modicon-Modbus-Protocol-design-violation
?to)
        (has-session-related-weakness-in-Modicon-Modbus-Protocol ?to)
        (has-susceptiblity-to-brute-force-attacks ?to)
        (has-unauthorized-access-to-PLC8 ?to)
        (has-access-to-plc-port-502-tcp ?to)
        (plc-compromised ?to))))
```

Fig. 3: A PDDL (Domain) representation of Attack Graph of Flare System.

```
(define (domain flare-fault-tree-domain)
(:requirements :strips :typing)
(:types node)
(:predicates
    (has-fault-mechanical-failure-due-to-compromised ?node - node)
    (has-fault-pcv-faulty-due-to-compromised ?node - node)
    (has-fault-flare-flameout)...)
(:action causes-fault-pcv-faulty-due-to-mechanical-failure
    :parameters (?to - node)
    :precondition (and
        (has-fault-mechanical-failure-due-to-compromised ?to))
    :effect (and
        (has-fault-pcv-faulty-due-to-compromised ?to)))
.
.
.
(:action causes-fault-flare-flameout
    :parameters (?node1 ?node2 - node)
    :precondition (and
        (has-fault-flame-detachment-due-to-compromised ?node1)
        (has-fault-pilot-extinction-due-to-compromised ?node2))
    :effect (and
        (has-fault-flare-flameout))))
```

Fig. 4: A PDDL (Domain) representation of Fault Tree of Flare System.

```
(define (problem flare-attack-graph-problem)
  (:domain flare-attack-graph-domain)
  (:objects
    Microsoft-Windows-12-Server firewall-vpn ... - node)
  (:init
    (has-compromised-customer-pc)
    (has-compromised-engineering-workstations) ...)
  (:goal
    (and (plc-compromised))))
```

Fig. 5: A PDDL (Problem) representation of Attack Graph of Flare System.

and goal state. Then the domain file (Fig. 7) of the optimistic model is given as an input to the Algo. 1. As a specific set of vulnerabilities is responsible for each safety event occurrence (before changing it using *Resilience Path Learning Algorithm* (RPLA)), it iteratively starts to change the post-conditions of

```
(define (problem flare-fault-tree-problem)
   (:domain flare-fault-tree-domain)
   (:objects
     plc-Allen-Bradley-MicroLogix-1100 plc-Siemens-S7-1200 ... - node)
   (:init
     (has-fault-mechanical-failure-due-to-compromised) ...)
   (:goal
     (and (has-fault-flare-flameout))))
```

Fig. 6: A PDDL (Problem) representation of Fault Tree of Flare System.

the actions based on the fault it triggers and removing all extra faults from that action as shown in Fig. 9. Once all the post-conditions of the actions to reach a specific goal ($d_1 \in D$) states from initial ($s_1 \in S$) states are updated by RPLA, a plan for resilience path ($p_{r1} \in P_R$) is created. By combining all the resilience paths ($p_{r1}, p_{r2} \dots p_{rk} \in P_R$) for all possible initial ($s_1, s_2 \dots s_k \in S$) and goal ($d_1, d_2 \dots d_k \in D$) states we can get the *Resiliency Graph* (RG). Fig. 10 illustrated the domain file of the final RG. An overall visual representation of *Attack Graph* (a), *Fault Tree* (b), and the corresponding *Resiliency Graph* (c) for Flare System has been illustrated in Fig. 14 (appendix).

C. Cascading Failure Analyses with RG

A cascading safety event in the context of CPS is a series of malfunctions or failures in a system that spread through connected components resulting in a catastrophic system failure. Fig. 11 represents a snapshot of such a scenario in the Flare System represented by the red line. When an attacker compromises a customer PC in the Flare System, it also gets access to the Microsoft Windows 12 server. By compromising a series of vulnerabilities such as CVE-2019-0575, CVE-2019-0584, CVE-2018-0538, and CVE-2017-9312, the attacker manages to take control of PLC Allen-Bradley ControlLogix which is responsible for controlling the isolation valve. Once the PLC Allen-Bradley gets compromised, it triggers a series of cascading safety events in the Flare System. From Table 1 it is evident that by compromising PLC Allen-Bradley ControlLogix, an attacker could make the PLC go offline (DoS attack). But previous researchers did not address the consequences when the PLC goes offline. At first, it closes the valve block in the CPS. Next, it turns off the manual isolation valve followed by a low-pressure supply in the pilot and finally the extinction of the pilot. Using our *Resilience Path Learning Algorithm* (RPLA) this cascading effect of failures can be captured by *Resiliency Graph* (RG) as shown in Fig. 12 (as plan for that resilience path). Not only this, but for every possible cyber-attack on the OT system, RG is able to express the relation between an attack and its corresponding faults (a cascading failure) in terms of a resilience path (or plan). This allows the CPS operators to better analyze and understand the cascading effect of a safety hazard that has not been addressed in earlier research.

VII. Evaluation

Our primary focus is to evaluate how the performance characteristics of *Resiliency Graph* (RG) changed with respect

```
(define (domain flare-domain-all)
(:requirements :strips :typing)
(:types node)
(:predicates
    (has-compromised-customer-pc)
    (has-compromised-engineering-workstations)
    (has-connected ?at - node ?to - node)
    (plc-offline ?node - node)
    (plc-compromised ?node - node)...)
(:action causes-fault-mechanical-failure-due-to
    :parameters (?to - node)
    :precondition (and
        (has-vulnerability-CVE-2017-6032 ?to)
        (has-Schneider-Electric-Modicon-Modbus-Protocol-design-violation ?to)
        (has-session-related-weakness-in-Modicon-Modbus-Protocol ?to)
        (has-susceptiblity-to-brute-force-attacks ?to)
        (has-unauthorized-access-to-PLC8 ?to)
        (has-access-to-plc-port-502-tcp ?to)
        (plc-compromised ?to))
    :effect (and
        (has-fault-mechanical-failure-due-to-compromised ?to)
        (has-fault-instrumental-failure-due-to-compromised ?to)
        (has-fault-pcv-faulty-due-to-compromised ?to)
        (has-fault-valve-blocked-close-due-to-compromised ?to)
        (has-fault-opertor-fault-due-to-compromised ?to)
        (has-fault-manual-isolation-valve-close-due-to-compromised ?to)
        (has-fault-fg-interrupted-at-source-due-to-compromised ?to)
        (has-fault-isolation-of-fg-line-for-works-due-to-compromised ?to)
        (has-fault-no-flow-of-fuel-gas-due-to-compromised ?to)
        (has-fault-failure-on-ignition-system-due-to-compromised ?to)
        (has-fault-ignition-pipe-clogged-due-to-compromised ?to)
        (has-fault-defect-on-ignition-system-due-to-compromised ?to)
        (has-fault-pilot-low-supply-pressure-due-to-compromised ?to)
        (has-fault-nitrogen-valve-open-due-to-compromised ?to)
        (has-fault-condensate-presence-in-fg-due-to-compromised ?to)
        (has-fault-pipe-not-drained)
        (has-fault-pilot-supply-pipe-isolated-due-to-compromised ?to)
        (has-fault-relief-pcv-closed-to-compromised ?to)
        (has-fault-switching-to-another-flare-due-to-compromised ?to)
        (has-fault-low-flow-gas-flaring-due-to-compromised ?to)
        (has-fault-pumping-phenomenon-due-to-compromised ?to)
        (has-fault-windspeed-greater-than-120-km-per-hr-due-to-compromised
?to)
        (has-fault-flame-detachment-due-to-compromised ?to)
        (has-fault-pilot-extinction-due-to-compromised ?to)
        (has-fault-flare-flameout)))
.
.
.
(:action causes-fault-pcv-faulty-due-to-mechanical-failure
    :parameters (?to - node)
    :precondition (and
        (has-fault-mechanical-failure-due-to-compromised ?to))
    :effect (and
        (has-fault-mechanical-failure-due-to-compromised ?to)
        (has-fault-instrumental-failure-due-to-compromised ?to)
        (has-fault-pcv-faulty-due-to-compromised ?to)
        (has-fault-valve-blocked-close-due-to-compromised ?to)
        (has-fault-opertor-fault-due-to-compromised ?to)
        (has-fault-manual-isolation-valve-close-due-to-compromised ?to)
        (has-fault-fg-interrupted-at-source-due-to-compromised ?to)
        (has-fault-isolation-of-fg-line-for-works-due-to-compromised ?to)
        (has-fault-no-flow-of-fuel-gas-due-to-compromised ?to)
        (has-fault-failure-on-ignition-system-due-to-compromised ?to)
        (has-fault-ignition-pipe-clogged-due-to-compromised ?to)
        (has-fault-defect-on-ignition-system-due-to-compromised ?to)
        (has-fault-pilot-low-supply-pressure-due-to-compromised ?to)
        (has-fault-nitrogen-valve-open-due-to-compromised ?to)
        (has-fault-condensate-presence-in-fg-due-to-compromised ?to)
        (has-fault-pipe-not-drained)
        (has-fault-pilot-supply-pipe-isolated-due-to-compromised ?to)
        (has-fault-relief-pcv-closed-to-compromised ?to)
        (has-fault-switching-to-another-flare-due-to-compromised ?to)
        (has-fault-low-flow-gas-flaring-due-to-compromised ?to)
        (has-fault-pumping-phenomenon-due-to-compromised ?to)
        (has-fault-windspeed-greater-than-120-km-per-hr-due-to-compromised
?to)
        (has-fault-flame-detachment-due-to-compromised ?to)
        (has-fault-pilot-extinction-due-to-compromised ?to)
        (has-fault-flare-flameout))))
```

Fig. 7: A PDDL (Domain) representation of overall Flare System before learning the P_R.

to the complexity of network (such as number of nodes, faults & vulnerabilities). We developed various experimental test scenarios to evaluate RG's performance. We ran our experiment on a 2.21 GHz Octa-Core Intel Core i7 processor, with 32 GB 2208 MHz, and LPDDR3 memory.

```
(define (problem flare-problem)
   (:domain flare-domain-all)
   (:objects
     Microsoft-Windows-12-Server firewall-vpn ... - node)
   (:init
     (has-compromised-customer-pc)
     (has-compromised-engineering-workstations) ...
   (:goal
     (and (has-fault-pilot-extinction-due-to-compromised))))
```

Fig. 8: A PDDL (Problem) representation of overall Flare System before learning the P_R.

```
(:action causes-fault-mechanical-failure-due-to
    :parameters (?to - node)
    :precondition (and
        (has-vulnerability-CVE-2017-6032 ?to)
        (has-Schneider-Electric-Modicon-Modbus-Protocol-design-violation ?to)
        (has-session-related-weakness-in-Modicon-Modbus-Protocol ?to)
        (has-susceptiblity-to-brute-force-attacks ?to)
        (has-unauthorized-access-to-PLC8 ?to)
        (has-access-to-plc-port-502-tcp ?to)
        (plc-compromised ?to))
    :effect (and
        (has-fault-mechanical-failure-due-to-compromised ?to)
        ~~(has-fault-instrumental-failure-due-to-compromised ?to)~~
        ~~(has-fault-pcv-faulty-due-to-compromised ?to)~~
        ~~(has-fault-valve-blocked-close-due-to-compromised ?to)~~
        ~~(has-fault-opertor-fault-due-to-compromised ?to)~~
        ~~(has-fault-manual-isolation-valve-close-due-to-compromised ?to)~~
        ~~(has-fault-fg-interrupted-at-source-due-to-compromised ?to)~~
        ~~(has-fault-isolation-of-fg-line-for-works-due-to-compromised ?to)~~
        ~~(has-fault-no-flow-of-fuel-gas-due-to-compromised ?to)~~
        ~~(has-fault-failure-on-ignition-system-due-to-compromised ?to)~~
        ~~(has-fault-ignition-pipe-clogged-due-to-compromised ?to)~~
        ~~(has-fault-defect-on-ignition-system-due-to-compromised ?to)~~
        ~~(has-fault-pilot-low-supply-pressure-due-to-compromised ?to)~~
        ~~(has-fault-nitrogen-valve-open-due-to-compromised ?to)~~
        ~~(has-fault-condensate-presence-in-fg-due-to-compromised ?to)~~
        ~~(has-fault-pipe-not-drained)~~
        ~~(has-fault-pilot-supply-pipe-isolated-due-to-compromised ?to)~~
        ~~(has-fault-relief-pcv-closed-to-compromised ?to)~~
        ~~(has-fault-switching-to-another-flare-due-to-compromised ?to)~~
        ~~(has-fault-low-flow-gas-flaring-due-to-compromised ?to)~~
        ~~(has-fault-pumping-phenomenon-due-to-compromised ?to)~~
        ~~(has-fault-windspeed-greater-than-120-km-per-hr-due-to-compromised
?to)~~
        ~~(has-fault-flame-detachment-due-to-compromised ?to)~~
        ~~(has-fault-pilot-extinction-due-to-compromised ?to)~~
        ~~(has-fault-flare-flameout)~~))
```

Fig. 9: Updating actions of the Optimistic Model using RPLA.

To evaluate the time complexity of *Resilience Path Learning Algorithm* (RPLA), we converted the Flare System into an optimistic model with 11 nodes, 40 vulnerabilities, and 24 faults (visualization of RG shown in Fig. 14). For the analysis, we began with a set of initial states which is true for the Flare System, and set a goal such that it triggers a fault in the Flare System. Hence, for the investigation, we set the goal `(has-fault-flare-flameout)` to flame out the Flare System. The RPLA algorithm needed to update 10 actions within the Flare system. To do this the algorithm required 39 iterations and 2837.364 microseconds to find the relationships between vulnerabilities and the potential safety events i.e. flaming out the Flare System within the OT network.

We are generating test networks using Barabási–Albert preferential attachment [53] with random connections, number of vulnerabilities, and faults to assess the performance of RG. Fig. 13 shows the results of different network graphs in which we tested our algorithm. The graphs show that as network size increases the algorithm needs more time (in milliseconds) to discover the relationships between vulnerabilities and faults for all configurations. This is because, in comparison to smaller networks with fewer nodes, vulnerabilities, and faults, larger networks are required to make this connection which proves that our Algo. 1 is working consistently. On the other hand, the complexity of the goal has a big impact on how many iterations are needed. If achieving a goal (i.e., triggering specific faults) requires multiple actions to be adjusted, the algorithm needs

```
(define (domain flare-domain)
 (:requirements :strips :typing)
 (:types node)
 (:predicates
     (has-compromised-customer-pc)
     (has-compromised-engineering-workstations)
     (has-connected ?at - node ?to - node)
     (plc-offline ?node - node)
     (plc-compromised ?node - node)...)
 (:action causes-fault-mechanical-failure-due-to
     :parameters (?to - node)
     :precondition (and
         (has-vulnerability-CVE-2017-6032 ?to)
         (has-Schneider-Electric-Modicon-Modbus-Protocol-design-violation ?to)
         (has-session-related-weakness-in-Modicon-Modbus-Protocol ?to)
         (has-susceptiblity-to-brute-force-attacks ?to)
         (has-unauthorized-access-to-PLC8 ?to)
         (has-access-to-plc-port-502-tcp ?to)
         (plc-compromised ?to))
     :effect (and
         (has-fault-mechanical-failure-due-to-compromised ?to)))
 (:action causes-fault-pcv-faulty-due-to-mechanical-failure
     :parameters (?to - node)
     :precondition (and
         (has-fault-mechanical-failure-due-to-compromised ?to))
     :effect (and
         (has-fault-pcv-faulty-due-to-compromised ?to)))
 (:action causes-fault-pilot-low-supply-pressure-due-to-pressure-to-pcv-faulty
     :parameters (?to - node)
     :precondition (and
         (has-fault-pcv-faulty-due-to-compromised ?to))
     :effect (and
         (has-fault-pilot-low-supply-pressure-due-to-compromised ?to)))
 .
 .
 .
 (:action causes-fault-pilot-extinction-due-to-pilot-low-supply-pressure
     :parameters (?to - node)
     :precondition (and
         (has-fault-pilot-low-supply-pressure-due-to-compromised ?to))
     :effect (and
         (has-fault-pilot-extinction-due-to-compromised ?to))))
```

Fig. 10: A PDDL (Domain) representation of overall Flare System after learning the P_R and generating RG.

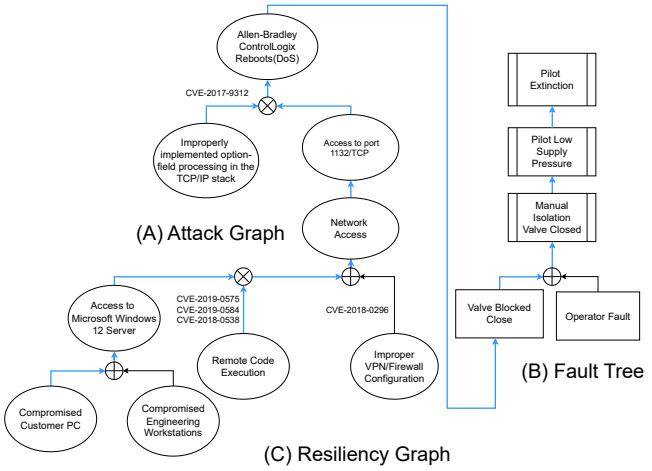


Fig. 11: Visual representation of a Resilience Path (P_R) capturing cascading failure with RG where $\bigoplus$ is "or" condition & $\bigotimes$ is "and" condition, and the blue line represented a resilience path $p_r \in P_R$.

```
(access-to-windows-server)
(exploits-vulnerability-CVE-2019-0575)
(exploits-vulnerability-CVE-2017-9312)
(causes-fault-valve-blocked-close-due-to)
(causes-fault-manual-isolation-valve-close-due-to-valve-blocked-close)
(causes-fault-pilot-low-supply-pressure-due-to-manual-isolation-valve-close)
(causes-fault-pilot-extinction-due-to-pilot-low-supply-pressure)
; cost = 6 (unit cost)
```

Fig. 12: Plan generated for Cascading Failures captured by RPLA for Fig. 11

more iterations to learn the *Resilience Path* (P_R). As we can see from Fig. 13(a), even though the network of node = 50, vulnerabilities = 50, and faults = 25 is larger compared to a network containing node = 35, vulnerabilities = 35, and faults = 17 it takes less number of iterations to learn the P_R as the goal state might have takes less number of actions to be modified. Thus, from the results, it can be concluded that our algorithm is working consistently that is the execution time is directly proportional to the network size. Additionally, it can be also observed that the time takes to learn P_R heavily depends on the safety events it triggers.

VIII. Conclusion

The *Resiliency Graph* (RG) framework aims to give CPS operators a tool, and insights needed to better evaluate and analyze how system vulnerabilities could cause safety issues and cascading failures. By revealing the intricate interdependencies between cyber threats, system vulnerabilities, and safety hazards, the RG gives a complete framework to identify, prioritize, and mitigate potential threats to the integrity and reliability of industrial control systems. This approach enhances the overall safety and reliability posture of CPS, ensuring continuous and secure operations in industrial environments. However, the current *Resilience Path Learning Algorithm* (RPLA) is limited in that it can only identify a single path, P_R, between the initial and goal states, rather than all possible paths within the system. Future work aims to improve this algorithm to identify all P_R paths for a given model, thereby further strengthening the system's resilience.

Acknowledgement

This work was partially supported by the U.S. National Science Foundation under Grant No. 1822118 and 2226232, Award Numbers DMS 2123761, the member partners of the NSF IUCRC Center for Cyber Security Analytics and Automation – AMI, NewPush, Cyber Risk Research, NIST and ARL – the State of Colorado (grant #SB 18-086) and the authors' institutions. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or other organizations and agencies.

References

[1] R. K. Thomas, M. C. Hatfield, I. Lozano, E. Overly, J. G. Korb, and J. Vu, "System and method for modeling and analyzing the impact of cyber-security events on cyber-physical systems," United States of America Patent US-10 262 143-B2, 2019.

[2] M. Roberts, A. E. Howe, I. Ray, and M. Urbanska, "Using Planning for a Personalized Security Agent," in *Workshop on Problem Solving using Classical Planners on Artificial Intelligence*, 2012.

[3] N. Ghosh and S. K. Ghosh, "A Planner-based Approach to Generate and Analyze Minimal Attack Graph," *Applied Intelligence*, vol. 36, pp. 369–390, 2012.

[4] B. Bezawada, I. Ray, and K. Tiwary, "AGBuilder: An AI Tool for Automated Attack Graph Building, Analysis, and Refinement," in *Data and Applications Security and Privacy XXXIII: Proceedings of the 33rd Annual IFIP WG 11.3 Conference*. Springer, 2019, pp. 23–42.

[5] K. Tiwary, S. Weerawardhana, I. Ray, and A. Howe, "PDDLAssistant: A Tool for Assisting Construction and Maintenance of Attack Graphs using PDDL," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS 2017)*, 2017.

[6] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated Generation and Analysis of Attack Graphs," in *Proceedings of 2002 IEEE Symposium on Security and Privacy*. IEEE, 2002, pp. 273–284.

[7] X. Ou, S. Govindavajhala, A. W. Appel *et al.*, "MulVAL: A Logic-Based Network Security Analyzer." in *Proceedings of the 14th USENIX Security Symposium*, vol. 8. Baltimore, MD, 2005, pp. 113–128.

[8] X. Ou, W. F. Boyer, and M. A. McQueen, "A Scalable Approach to Attack Graph Generation," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 336–345.

[9] K. Kaynar and F. Sivrikaya, "Distributed Attack Graph Generation," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 519–532, 2015.

[10] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," *IEEE Transaction on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2011.

[11] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal Security Hardening using Multi-objective Optimization on Attack Tree Models of Networks," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 204–213.

[12] R. Dewri, I. Ray, N. Poolsappasit, and D. Whitley, "Optimal Security Hardening on Attack Tree Models of Networks: A Cost-Benefit Analysis," *International Journal of Information Security*, vol. 11, no. 3, pp. 167–188, 2012.

[13] M. U. Aksu, K. Bicakci, M. H. Dilek, A. M. Ozbayoglu, and E. ı. Tatli, "Automated Generation of Attack Graphs Using NVD," in *Proceedings of the 18th ACM Conference on Data and Application Security and Privacy*, 2018, pp. 135–142.

[14] H. A. Watson *et al.*, "Launch Control Safety Study," *Bell Labs*, 1961.

[15] K. J. Sullivan, J. B. Dugan, and D. Coppit, "The Galileo Fault Tree Analysis Tool," in *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No. 99CB36352)*. IEEE, 1999, pp. 232–235.

[16] L. Xing and S. V. Amari, *Fault Tree Analysis*. Springer, 2008, pp. 595–620.

[17] K. Aslansefat and G.-R. Latif-Shabgahi, "A Hierarchical Approach for Dynamic Fault Trees Solution through Semi-Markov Process," *IEEE Transactions on Reliability*, vol. 69, no. 3, pp. 986–1003, 2020.

[18] C. Phillips and L. P. Swiler, "A Graph-based System for Network-vulnerability Analysis," in *Proceedings of the 1998 New Security Paradigms Workshop*. Association for Computing Machinery, 1998, p. 71–79.

[19] H. R. Shahriari and R. Jalili, "Vulnerability Take Grant (VTG): An Efficient Approach to Analyze Network Vulnerabilities," *Computers and Security*, vol. 26, no. 5, pp. 349–360, 2007.

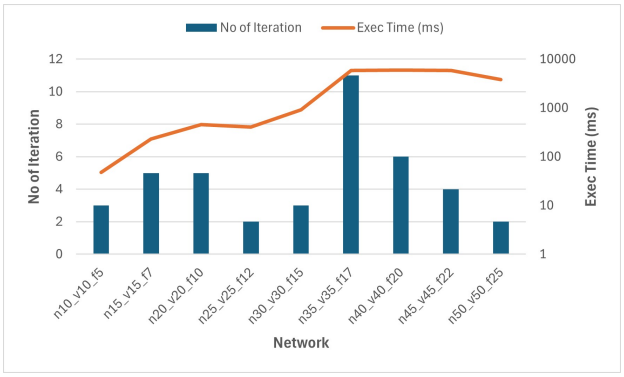
[20] O. Sheyner and J. Wing, "Tools for Generating and Analyzing Attack Graphs," in *Proceedings of the International Symposium on Formal Methods for Components and Objects*. Springer, 2003, pp. 344–371.

[21] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A New Symbolic Model Checker," *International Journal oo Software Tools for Technology Transfer*, vol. 2, pp. 410–425, 2000.

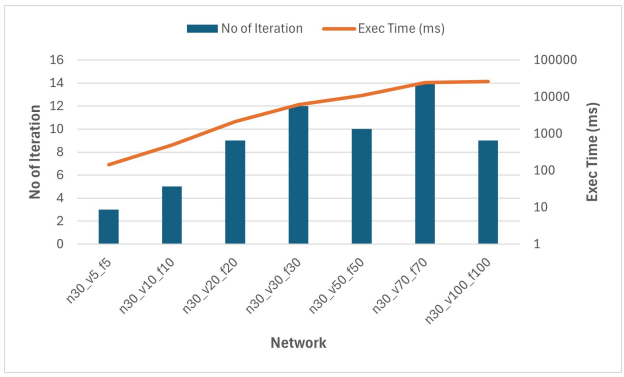
[22] C. Hankin, P. Malacaria *et al.*, "Attack Dynamics: An Automatic Attack Graph Generation Framework Based on System Topology, CAPEC, CWE, and CVE Databases," *Computers and Security*, vol. 123, p. 102938, 2022.

[23] S. Jajodia, S. Noel, and B. O'berry, "Topological Analysis of Network Attack Vulnerability," *Managing Cyber Threats: Issues, Approaches, and Challenges*, pp. 247–266, 2005.

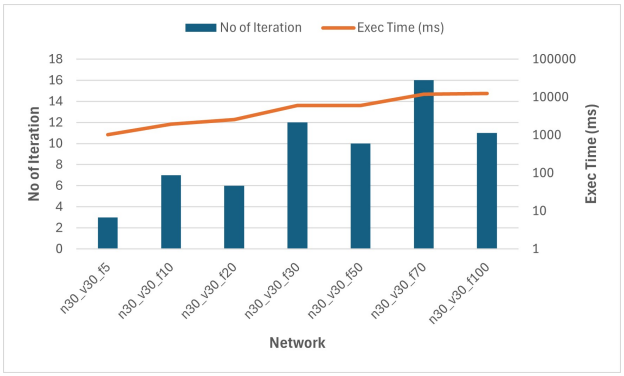
[24] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical Attack Graph Generation for Network Defense," in *Proceedings of the 22nd Annual*



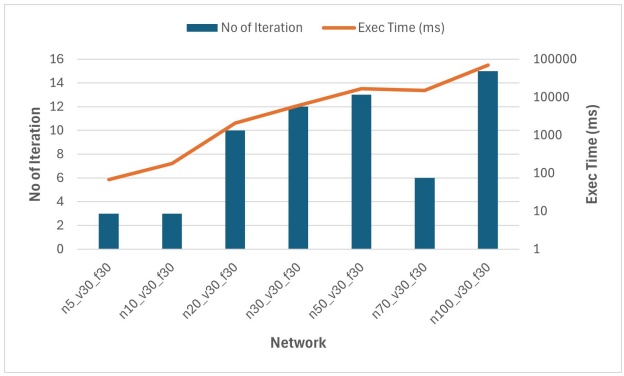
(a) No. of Faults varied keeping Nodes and Vulnerabilities constant.



(b) No. of Vulnerabilities and Faults varied keeping Nodes constant at 30.



(c) No. of Faults varied keeping Nodes and Vulnerabilities constant at 30.



(d) No. of Nodes varied keeping Vulnerabilities and Faults constant at 30.

Fig. 13: No. of Iterations and Time required by the RPLA to identify the connections between AG Vulnerabilities and the Faults they can trigger in the FT across different network configurations. It can be observed that larger networks require more time but not necessarily more iterations, depending on the complexity of the desired fault to be triggered in the system.

Computer Security Applications Conference (ACSAC'06). IEEE, 2006, pp. 121–130.

[25] J. Lee, D. Moon, I. Kim, and Y. Lee, "A Semantic Approach to Improving Machine Readability of a Large-scale Attack Graph," *Journal of Supercomputing*, vol. 75, pp. 3028–3045, 2019.

[26] S. Noel and S. Jajodia, "Managing Attack Graph Complexity through Visual Hierarchical Aggregation," in *Proceedings of the 2004 ACM Workshop on Visualization and Data mining for Computer Security (VizSEC/DMSEC '04)*, 2004, pp. 109–118.

[27] J. Hong and D.-S. Kim, "HARMS: Hierarchical Attack Representation Models for Network Security Analysis," in *Proceedings of the 2012 Australian Information Security Management Conference*. Edith Cowan University, 2012.

[28] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing, "Ranking Attack Graphs," in *Proceedings of the 2006 International Workshop on Recent Advances in Intrusion Detection*. Springer, 2006, pp. 127–144.

[29] S. J. Templeton and K. Levitt, "A Requires/provides Model for Computer Attacks," in *Proceedings of the 2000 New Security Paradigms Workshop*, 2001, pp. 31–38.

[30] M. Ibrahim, Q. Al-Hindawi, R. Elhafiz, A. Alsheikh, and O. Alquq, "Attack Graph Implementation and Visualization for Cyber Physical Systems," *Processes 2020*, vol. 8, no. 1, p. 12, 2019.

[31] S. Wang, Z. Zhang, and Y. Kadobayashi, "Exploring Attack Graph for Cost-benefit Security Hardening: A Probabilistic Approach," *Computers and Security*, vol. 32, pp. 158–169, 2013.

[32] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An Attack Graph-based Probabilistic Security Metric," in *Data and Applications Security XXII: 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*. Springer, 2008, pp. 283–296.

[33] B. Kordy, P. Kordy, S. Mauw, and P. Schweitzer, "Adtool: Security Analysis with Attack–Defense Trees," in *Proceeding of the 10th International Conference on the Quantitative Evaluation of Systems (QEST)*. Springer, 2013, pp. 173–176.

[34] H. S. Lallie, K. Debattista, and J. Bal, "An Empirical Evaluation of the Effectiveness of Attack Graphs and Fault Trees in Cyber-Attack Perception," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1110–1122, 2017.

[35] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, Graph-based Network Vulnerability Analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 217–224.

[36] M. Khouzani, Z. Liu, and P. Malacaria, "Scalable Min-max Multi-objective Cyber-security Optimisation over Probabilistic Attack Graphs," *European Journal of Operational Research*, vol. 278, no. 3, pp. 894–903, 2019.

[37] M. Čepin and M. Čepin, "Event Tree Analysis," *Assessment of Power System Reliability: Methods and Applications*, pp. 89–99, 2011.

[38] J. D. Andrews and S. J. Dunnett, "Event-tree Analysis using Binary Decision Diagrams," *IEEE Transactions on Reliability*, vol. 49, no. 2, pp. 230–238, 2000.

[39] R. Kenarangui, "Event-tree Analysis by Fuzzy Probability," *IEEE Transactions on Reliability*, vol. 40, no. 1, pp. 120–124, 1991.

[40] J. B. Bowles and C. E. Peláez, "Fuzzy Logic Prioritization of Failures in a System Failure Mode, Effects and Criticality Analysis," *Reliability Engineering and System Safety*, vol. 50, no. 2, pp. 203–213, 1995.

[41] H.-C. Liu, L. Liu, and N. Liu, "Risk Evaluation Approaches in Failure Mode and Effects Analysis: A Literature Review," *Expert Systems with Application*, vol. 40, no. 2, pp. 828–838, 2013.

[42] P. Crucitti, V. Latora, and M. Marchiori, "Model for Cascading Failures in Complex Networks," *Physical Review E*, vol. 69, no. 4, p. 045104, 2004.

[43] H. Wang, Z. Chen, J. Zhao, X. Di, and D. Liu, "A Vulnerability

Assessment Method in Industrial Internet of Things Based on Attack Graph and Maximum Flow," *IEEE Access*, vol. 6, pp. 8599–8609, 2018.
[44] I. Ray, S. Sreedharan, R. Podder, S. K. Bashir, and I. Ray, "Explainable AI for Prioritizing and Deploying Defenses for Cyber-Physical System Resiliency," in *Proceedings 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications*, 2023, pp. 184–192.
[45] N. Ghosh, "A Planner-based Approach to Generate and Analyze Minimal Attack Graph," *Applied Intelligence*, vol. 36, pp. 369–390, 2012.
[46] Z. Yichao, Z. Tianyang, G. Xiaoyue, and W. Qingxian, "An Improved Attack Path Discovery Algorithm through Compact Graph Planning," *IEEE Access*, vol. 7, pp. 59 346–59 356, 2019.
[47] C. Sarraute, "Automated Attack Planning," *arXiv preprint arXiv:1307.7808*, 2013.
[48] M. S. Boddy, J. Gohde, T. Haigh, and S. A. Harp, "Course of Action Generation for Cyber Security using Classical Planning," in *Proceedings of the 15th International Conference on International Conference on Automated Planning and Scheduling*, 2005, pp. 12–21.
[49] M. Roberts, A. Howe, I. Ray, M. Urbanska, Z. S. Byrne, and J. M. Weidert, "Personalized Vulnerability Analysis through Automated Planning," in *Working Notes for the 2011 IJCAI Workshop on SecArt*, 2011, p. 50.
[50] S. Sreedharan and M. Katz, "Optimistic Exploration in Reinforcement Learning Using Symbolic Model Estimates," in *Advances in Neural Information Processing Systems*, vol. 36. Curran Associates, Inc., 2023, pp. 34 519–34 535.
[51] M. T. Berrouane and Z. Lounis, "Safety Assessment of Flare Systems by Fault Tree Analysis," *Journal of Chemical Technology and Metallurgy*, vol. 51, no. 2, pp. 229–234, 2016.
[52] S. Kabir, M. Taleb-Berrouane, and Y. Papadopoulos, "Dynamic Reliability Assessment of Flare Systems by Combining Fault Tree Analysis and Bayesian Networks," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, vol. 45, no. 2, pp. 4305–4322, 2023.
[53] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

APPENDIX

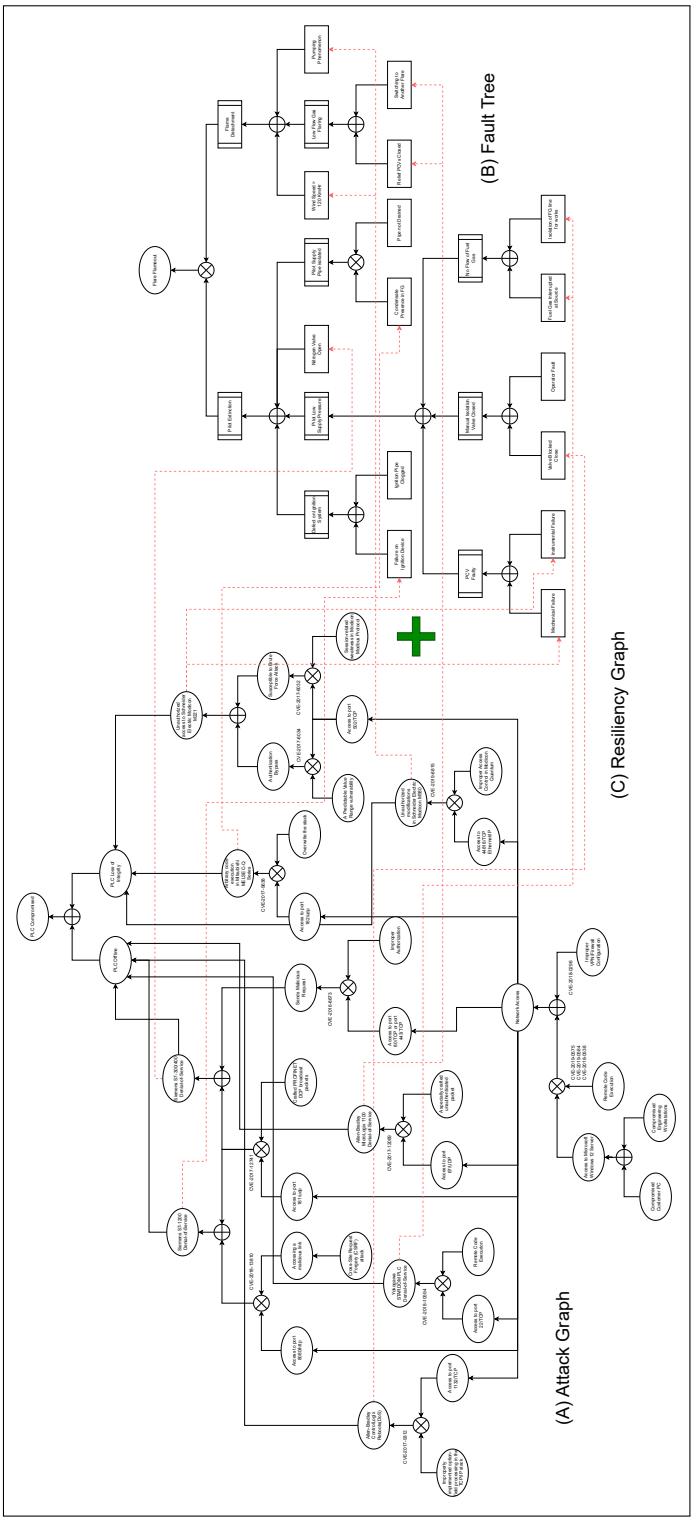


Fig. 14: Resiliency Graph for LNG Complex Flare System of Figure 2 (a), modeling safety issues via fault trees (b). Resiliency Graphs (c) allow us to compose (+ represents composition, $\bigoplus$ is "or" condition & $\bigotimes$ is "and" condition) (a) and (b) into a single unified representation. The dotted lines represent the areas where security and safety need to be composed.