Chaos Engineering for Enhanced Resilience of Cyber-Physical Systems

Charalambos Konstantinou, *Senior Member, IEEE*, George Stergiopoulos, *Member, IEEE*,
Masood Parvania, *Senior Member, IEEE*, and Paulo Esteves-Verissimo, *Fellow, IEEE*

Abstract—Cyber-physical systems (CPS) incorporate the complex and large-scale engineered systems behind critical infrastructure operations, such as water distribution networks, energy delivery systems, healthcare services, manufacturing systems, and transportation networks. Industrial CPS in particular need to simultaneously satisfy requirements of available, secure, safe and reliable system operation against diverse threats, in an adaptive and sustainable way. These adverse events can be of accidental or malicious nature and may include natural disasters, hardware or software faults, cyberattacks, or even infrastructure design and implementation faults. They may drastically affect the results of CPS algorithms and mechanisms, and subsequently the operations of industrial control systems (ICS) deployed in those critical infrastructures. Such a demanding combination of properties and threats calls for resilience-enhancement methodologies and techniques, working in real-time operation. However, the analysis of CPS resilience is a difficult task as it involves evaluation of various interdependent layers with heterogeneous computing equipment, physical components, network technologies, and data analytics. In this paper, we apply the principles of chaos engineering (CE) to industrial CPS, in order to demonstrate the benefits of such practices on system resilience. The systemic uncertainty of adverse events can be tamed by applying runtime CE-based analyses to CPS in production, in order to predict environment changes and thus apply mitigation measures limiting the range and severity of the event, and minimizing its blast radius.

Index Terms—Cyber-physical systems, industrial control systems, chaos engineering, resilience.

I. INTRODUCTION

The term cyber-physical systems (CPS) refers to systems which integrate and interconnect the physical environment with control, computing, and networking components. In the last decade, there has been a rapid growth of CPS in critical infrastructures including power systems, manufacturing facilities, robotics, transportation networks, desalination plants, and other industrial control systems (ICS).

These substantial applications of CPS, especially in industrial environments, have largely computerized critical infrastructures, making them subject to diverse threats which industrial CPS need to face, by simultaneously satisfying

C. Konstantinou and P. Esteves-Verissimo are with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia (e-mail: {charalambos.konstantinou, paulo.verissimo}@kaust.edu.sa).

G. Stergiopoulos is with the Department of Information and Communication Systems Engineering, University of the Aegean, Greece (e-mail: g.stergiopoulos@aegean.gr).

M. Parvania is with the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT 84112 USA (e-mail: masood.parvania@utah.edu).
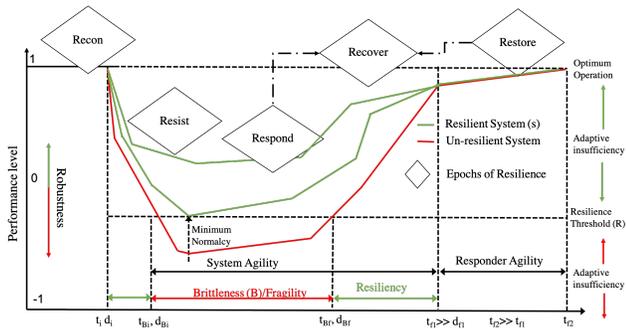
requirements of available, secure, safe and reliable system operation, in an adaptive and sustainable way.

These adverse events can be of accidental or malicious nature and may include natural disasters such as hurricanes, earthquakes, wildfires, hardware or software faults, e.g., in smart monitoring devices due to bugs in the code (e.g., operating system, compilers, libraries, etc.), or bit flips induced by cosmic rays, or 'silent errors' due to components and manufacturing variability faults, as well as cyberattacks, infrastructure design and implementation faults and, last but not least, unexpected operating conditions. They may drastically affect the results of CPS algorithms and mechanisms, and subsequently the operations of ICS deployed in those critical infrastructures, leading in the end to potential financial, social, legal, and political consequences.

Such a demanding combination of requirements versus threats calls for correspondingly powerful defenses, beyond classic industry-practice mitigation techniques. In other words, industrial CPS should seek *resilience*, or the continued ability to "endure" and "recover" from disrupting events. Resilient computing systems are ones that: have built-in baseline defences; can cope with virtually any quality of threat, be it accidental or malicious; protect in an incremental way and automatically respond and adapt to threats; and provide unattended and sustainable operation [1]. However, the cratfing and analysis of CPS resilience is a difficult task as it involves, as usual in this sector, various interdependent layers with heterogeneous computing equipment, physical components, network technologies, and data analytics.

In order to enhance and evaluate CPS resilience, there exist various works on resilience definitions, metrics, and evaluation and enhancement methods. A conceptual resilience curve with resilient control metrics is the "Disturbance and Impact Resilience Evaluation Curve" (DIRE), presented in Fig. 1, expressing the "Rs" of resilience, i.e., Recon, Resist, Respond, Recover, and Restore according to [2]. Experimentation practices involve the use of hardware-in-the-loop (HIL) testbed environments [3] and digital twins [4], Monte Carlo simulation-based methods [5], pre-selected scenario-based methods, and machine learning-based methods [6]. Existing methods, however, are often not examined in deployed CPS nor governed by a proactive experimentation practice in which weaknesses can be revealed before impacting the system, e.g., often HIL experimentation is specific to the application and scenarios under test [7], and not following systematic experimentation steps.

Fig. 1: Disturbance and impact resilience evaluation (DIRE) Curve (i=initial, f=final) [2].

Fig. 2: Chaos engineering (CE) experimentation principles.

In this work, we apply the principles of chaos engineering (CE) to industrial CPS, in order to demonstrate the benefits of such practices on system resilience. CE has been primarily used for software systems, and can be defined as "the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production" [8]. We propose to extend the concept to a more general framework including overall CPS architectures (hardware and software), as well as a broader scope of faults beyond software.

CE involves the facilitation of experiments to reveal weaknesses within a system, i.e., how the system responds to unexpected events such as outages due to cyberattacks or faults. The concept of CE was originated from Netflix in an effort to develop a built-in resilience tool within the production environment [9]. Typically, CE experiments include the following steps (Fig. 2): (1) define the 'steady state' of a system in order to quantify its characteristic behavior under a certain type of events; (2) build a hypothesis around the steady state which should be followed in both the control/theoretical aspect and the experiments (measurable output of a system); (3) vary process events into the systems reflecting realistic adverse productions conditions; (4) run automated experiments within the system production environment in an effort to disprove the hypothesis in terms of control and experiments differences.

This experimentation process builds confidence in the system with regards to its steady states of operation and how hard they are to be disrupted. In case CE experimentation identifies a systemic weakness, the overall process is set up in a way to ensure that potential impact radius from the experiment is minimized and contained. The systemic uncertainty of adverse events can thus be tamed by applying runtime CE-based analyses to CPS in production, in order to predict environment changes and thus apply mitigation measures limiting the range and severity of the event, and minimizing its blast radius[1].

In terms of quantitative benefits, CE can reduce the number

---

[1]Blast radius is often defined what is impacted with the CE tests and experiments. The effort to "minimize the blast radius" essentially refers to effort of identifying via CE experimentation the vulnerabilities causing failure "without accidentally blowing everything up" [10].
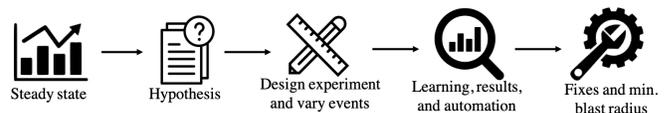
of preventable outages. This reduction of the blast radius is coupled with chaos-induced bounded harm that is revealed by CE experiments. Another aiding approach to CE is modeling outages using historical data, i.e., a proper implementation of CE experiments requires an establishment of a baseline for test metrics based on historical data. Taking proper records from an incident management system in place, and identifying vulnerabilities where the failure of a seemingly low-risk node (i.e., with subliminal criticality) could bring the system down leading to an outage, provides insight to quantitative financial loss due to the occurred incident as well as information about vulnerable nodes that should be injected in the next CE experiment. The adoption process may require engineers to mitigate risks of impacting customers with these experiments. This means there is a balance between reducing the impact to customers and identifying vulnerabilities quickly. As for qualitative benefits of CE, these include improvement in design decisions, growth in understanding service criticality, and confidence in validating reliability measures.

In this paper, we introduce the concept of CE for enhancing the resilience of CPS, and specifically, ICS environments of critical infrastructures. First, we review experimentation practices for weaknesses identification and testing as well as present existing CE approaches (Section II). We then build a case of a CE formal model in which we define and describe the control theoretical aspects of CE within a nonlinear cyber-physical ICS (Section III). Furthermore, we present a use-case scenario on the industrial process control model of the Tennessee Eastman Challenge in an effort to map the CE experimentation steps with actual metrics and failure events (Section IV). Finally, Section V concludes the paper and provides directions for future work.

II. RELATED WORK AND LIMITATIONS

Modern distributed systems and content delivery networks are often implemented as software-based services. Testing and vulnerability detection in distributed systems is a broad field that borrows numerous techniques from different engineering fields. CPS, and in particular modern ICS, are typically implemented as distributed software-based systems that control embedded devices and field sensors. Yet to our knowledge, there has been no prior use of CE experiments on ICS. As such, relevant publications cover two distinct areas, namely *(A)* vulnerability and fault diagnosis in ICS, and *(B)* current practices of CE in distributed systems.

A. Vulnerability Detection and Fault Testing in ICS

There exist several techniques for vulnerability detection and failure testing in software-based distributed systems that

share common characteristics with CE experiments. In software, existing approaches can be broadly classified into:

1) Static and dynamic analysis methods including fuzzing techniques. These approaches identify issues inside the code of embedded components and industrial systems, either offline (static) or during run-time execution of software (dynamic).
2) Fault and attack injection on ICS components.
3) Conditional testing approaches for the detection of system deviations, mostly implemented with invariant mining techniques.

There is extensive research on static analysis of industrial software for fault detection. Recent approaches utilize various techniques, such as regression verification [11], model checking [12], [13], or sequential function charts for the programming of the programmable logic controllers (PLCs). Similar approaches have extended these methods by including specifications of plant behavior converted to hybrid automata for verifying safety properties [14], [15].

Concerning dynamic analysis approaches, there exists work that delves into fault testing of ICS components. Authors in [16] have presented how tools can generate test inputs through symbolic execution to detect program execution errors. Vet-PLC is another attempt that combines static and dynamic analysis presented in [17]. Its goal is to verify real-world PLC code driven by events and detect safety violations in code. The work combines different types of analysis to understand causal relations among events in PLC code and quantitatively gauge temporal dependencies that are constrained by machine operations.

Published research has also used fuzz testing in ICS, with existing work targeting both industrial protocols and embedded systems. For example, authors in [18] were able to fuzz test function codes in ICS protocols and extract semantic information for vulnerability detection. Other approaches opted to fuzz input commands on firmware binaries extracted from ICS to evaluate the security of specific libraries inside ICS components [19], [20], or inject large number of attacks on network protocols and monitor target systems and their responses for unexpected behavior that might suggest the existence of vulnerabilities [21], [22].

Mining invariant values from ICS devices has also been in scope of recent research in fault detection of ICS software. Recent approaches aim to extract operational conditions from system logs via data mining. These conditional variables are then evaluated whether they hold true at some point in time during varying executions of ICS processes [23], [24]. Other approaches mine such conditions from smart meters and medical devices, mostly for intrusion detection purposes [25]. Chen *et al.* extracted invariants from data traces of a water purification testbed to detect anomalies in system execution [26]. Other mixed approaches may also use invariants for vetting the source code of ICS components [17].

CE experiments try to deal with the fact that functional charts and specifications cannot capture the scale and complexity of user behavior in modern distributed systems. As such, CE focuses on experimenting on the entire system under test as a single entity. Even though such approaches are not able to formally define function executions, they have shown great promise in uncovering unwanted states of execution that were previously unknown and not depicted in functional specifications [9], [27]. CE experiments may also utilize fuzzing techniques for input, but they do not consider specific technical inputs or their fuzzed input does not target specific software or components. Instead, CE experiment view all ICS components and field devices as a single system and try to fuzz real-world, high-level input to observe what happens to the ICS boundary.

Also, another difference between CE experiments and some of the aforementioned fault analysis and testing techniques lies on the fact that CE experiments do not follow a binary assertion logic (as is the case with invariants or static analysis). Most fault testing and assessment methods assign pre-considered restrictions and evaluate software-based systems against these precondition to assert them as true or false. Instead, CE experiments focus in creating new execution data and detect system states that were previously unknown to operators [9].

B. Existing CE Approaches

Concerning existing CE experiments, the most famous case is Netflix, where engineers have taken CE as an approach to experimentally verify distributed systems' reliability. That is achieved either by changing the boundary state of components and analyzing system behavior using an internal service that was created specifically for his purpose (Chaos Monkey) [9], or by introducing false injections at the boundary of Netflix micro-services [27]. Chaos Monkey was also used in [28], where researchers proposed a balanced use of the service to inject variable degrees of failure into the network without disconnecting it and assess it against network invariant metrics.

Large-scale Java applications have also been known to utilize CE for testing complex distributed software [29]. Other implementations of the CE principles aimed to automatically inject system failures and errors into a containerized application to evaluate the overall system resilience under unknown states of operation [30]. Researchers in [31] detected faults in industrial network data using recurrent neural networks that incorporate CE principles to improve the efficiency of the tuning procedure. Neural networks and CE was also proposed in [32] as a means for generating unwanted states in nonlinear electrical circuits. Authors aimed to discover vulnerabilities on large, complex systems, such as the U.S. power grid.

Other research has focused on using CE for analyzing the execution states of Infrastructure-as-a-Service (IaaS) cloud platforms. For example, CloudStrike implemented the principles of CE by injecting faults to cloud resources to experiment on cybersecurity breaches caused by human errors and misconfigured resources [33], [34].

III. A Generic Chaos Engineering (CE) Model

The cyber-physical model of ICS can be seen as a model of a nonlinear system in the following state-space form:

$$\dot{x}(t) = g(x(t), u(t), \delta(t)) \qquad (1)$$
$$\Delta y(t) = h(x(t), u(t), \delta(t)) \qquad (2)$$

where the state $x(t) \in \mathbb{R}^n$ with $x(0) = x_0$, the measurement output deviations $\Delta y(t) \in \mathbb{R}^m$, the control input $u(t) \in \mathbb{R}^m$ (e.g., could represent set-points for states regulation), and $\mathcal{N} = \{1, \cdots, n\}$ represents the index set of the ICS components. $\delta(t) \in \mathbb{R}^{n_\delta}$ denotes the piecewise vector of constant signals including disturbances, measurement noise, and unknown control parameters. The model in Eqs. (1), (2) may describe water treatment facility, a desalination plant, a microgrid or any other cyber-physical ICS, and might have been derived under suitable regularity constraints from a more general differential-algebraic model [35].

1) Steady state definition in ICS environments: One major requirement of any CE experiment is to determine the system's steady state, i.e., that state in which the performance of the system is contributing towards maintaining a particular system property in a specific pattern or within a specific range. The objective in this part is the model development of the system in such a way to be able to describe the steady states according to the anticipated conditions of the system metrics. In the scenario of ICS, such metrics could be the level of sodium hydroxide in a water treatment plant [36].

Definition 1 (Steady State Stability and Synchronization). *There exist domains $\mathcal{V} \subseteq \mathbb{R}^m \times \mathbb{R}^{n_\delta}$ and $\mathcal{W} \subseteq \mathbb{R}^n$ for which:*
(1) g and h are Lipschitz continuous functions on $\mathcal{W} \times \mathcal{V}$,
(2) A Lipschitz continuous function ϕ_w on $\mathcal{W}$ is differentiable $\phi_w : \mathcal{V} \to \mathcal{W}$ and $\forall (u, \delta) \in \mathcal{V}$ satisfies $0 = g(\phi_w(u, \delta), u, \delta)$, i.e., each pair $(u, \delta) \in \mathcal{V}$ is unique on $\mathcal{W}$'s equilibrium state $\phi_w(u, \delta)$.
(3) There exist constants $c_j > 0$, $j \in \mathbb{N} : j \in [1, 4]$, and a continuously differentiable function f in (x, u), $f : \mathcal{W} \times \mathcal{V} \to \mathbb{R}_{\geq 0}$, $(x, (u, \delta)) \mapsto f(x, u, \delta)$ for which $\forall x \in \mathcal{W}$ and $\forall (u, \delta) \in \mathcal{V}$:

$$c_1 \|x - \phi_w(u, \delta)\|_2^2 \leq f(x, u, w) \leq c_2 \|x - \phi_w(u, \delta)\|_2^2$$
$$\nabla_x f(x, u, w)^\top f(x, u, \delta) \leq -c_3 \|x - \phi_w(u, \delta)\|_2^2$$
$$\|\nabla_u f(x, u, \delta)\|_2 < c_4 \|x - \phi_w(u, \delta)\|_2$$

(1)-(3) essentially present a singular perturbation problem in which relaxations and other variants exist, and (3) in particular, is a Lyapunov function determining exponential stability of $\phi_w(u, \delta) \in \mathcal{W}$ [37].
(4) The input-to-output equilibrium mapping $\Delta \bar{y}(u, \delta) = h(\phi_w(u, \delta), u, w) : \mathcal{V} \to \mathbb{R}^m$ has the form of:

$$\Delta \bar{y}(u, \delta) = h(\phi_w(u, \delta), u, w)$$
$$= (1/\gamma) * \mathbb{1}_m * (\mathbb{1}_m^\top \bar{u} - d) \qquad (3)$$

where $\gamma > 0$, $\mathbb{1}_m$ representing the indicator function of size m, and $d_{cu} \in \mathbb{R}$ is an unmeasured constant disturbance.
(4) shows that in steady state conditions, the synchronization of output measurements is possible with deviations being the same at all ICS components. The steady state of an output state can be seen as $\mathbb{1}_m^\top u - d$, e.g., the difference of load demand and generation control inputs in a power grid model. The linear span of the indicator function is chosen due to the objective of having finite linear combinations as indicator step functions on arbitrary intervals.

In ICS, system operators or automated functionalities of the ICS allocate actions to the actuating elements of the systems in an effort to operate across nominal system setpoint metrics. The target setpoints can be determined in a minimization-type of problem:

$$\begin{array}{ll} \underset{u_h \in \mathbb{R}^2}{\text{minimize}} & z(u_h) := \sum_{j \in V} z_j(u_{hj}) \\ \text{s.t.} & 0 = \mathbb{1}_m^\top u - d_{cu} \end{array} \qquad (4)$$

where $j \in V$ and $|V| = m$ is the set of the system nodes. Eq. (4) is assumed to be strictly feasible, i.e., the constraint is satisfied. $z_j : \mathcal{U}_i \to \mathbb{R}$ models the disutility of the j_{th} ICS component of not satisfying system nominal demands, and encompasses a barrier function to enforce inequalities $u_{h_j} \in \mathcal{U}_j = (u_{l_j}, u_{h_j})$, where $-\infty \leq u_{l_j} < u_{h_j} \leq +\infty$. The constraint of Eq. (4) ensures balance of system control inputs $\mathbb{1}_m^T u$ and unmeasured distrubances d_{cu}. In addition, by Eq. (3), Eq. (4) ensures that the steady state system deviations should lead to zero. An example metric following the last minimization problem could be defined as the frequency regulation setpoint of secondary frequency control of ICS in power grids.

2) Formulating hypotheses: Once the metrics are determined and the steady state behavior of the ICS is understood, the experimental hypotheses can be defined. The objective is to determine what is expected as the result of an experiment, in terms of the system's steady state conditions, if we apply a set of diverse events into the ICS. Lemma 2 relies on the minimization problem of Eq. (4) to determine under some assumptions the outcome of the experiment in terms of the target setpoints.

Definition 2 (Direct Graph and Connectivity). *A sensor network of an ICS can be seen as a topological map represented by a directed graph $\mathcal{G} = (V, E, A)$ with nodes $V = \{1, \ldots, m\}$, edges $E \subseteq V \times V$, and $A = [a_{ij}] \in \mathbb{R}^{m \times m}$ is a weighted adjacency matrix with non-negative adjacency elements, i.e., $a_{ij} \geq 0$, and $a_{ij} > 0$ if and only if the edge $(i, j) \in E$. The neighbors of node i are defined as $\mathcal{N}_i \triangleq \{j \in V : (i, j) \in E\}$.*

Definition 3 (Laplacian Matrix). *The Laplacian matrix $L \in \mathbb{R}^{m \times m}$ of the graph $\mathcal{G}$ of Definition 2 is defined by its elements given by:*

$$L_{ij} = \begin{cases} -a_{ij} & \text{if } i \neq j \\ \sum_{k \neq i} a_{ik} & \text{if } i = j \end{cases}$$

Every row sum of L is zero. As for its eigenvalues λ, by definition there exist $\lambda_j = 0$ with right-eigenvector $X_{R_j} = \mathbb{1}_m$ and the rest λ_i, $i \neq j$, have positive real part [38].

Definition 4 (Globally Reachable Node). *A node $v_i \in \mathrm{V}$ is a globally reachable node, if there exists a path (i.e., a combination of edges in E) from it to every other nodes in $\mathcal{G}$.*

The next lemma shows an important property of Laplacian L.

Lemma 1. *The graph $\mathcal{G}$ has a globally reachable node if and only if the Laplacian L of $\mathcal{G}$ has a simple zero eigenvalue. In this scenario, the left-eigenvector $\mathrm{X_{L_j}} \in \mathbb{R}^m$ of L associated with the simple $\lambda_j = 0$ is $\mathrm{X_{L_j}} \geq 0$, and $\mathrm{X_{L_j}} > 0$ if and only if node $v_j \in \mathrm{V}$ is globally reachable.*

Lemma 2. *Consider Definition 2 of a sensor network of an ICS represented by a weighted directed graph $\mathcal{G} = (\mathrm{V}, \mathrm{E}, \mathrm{A})$ with Laplacian matrix L, and let such graph to follow also Definition 4, i.e., include a globally reachable node and the left-eigenvector $\mathrm{X_{L_j}} \in \mathbb{R}^m_{\geq 0}$ of L to correspond to its simple eigenvalue $\lambda_j = 0$. Let $\bar{\Delta y}$ follows the form of Eq. (3) and $u_h \in \mathbb{R}^m$, then for the case of a diagonal matrix $D \succeq 0$ for which $\mathrm{X_{L_j}}^\top D \mathbb{1}_m > 0$, there exist the following equivalent statements: (1) u_h is the unique optimal solution of Eq. (4); and (2) the vector $q_h \in \mathrm{span}\,(\mathbb{1}_m)$ is unique and satisfies:*

$$0 = D\bar{\Delta y}(u_h, \mathrm{X_{L_j}}) + \mathrm{L}q_h \tag{5a}$$
$$u_h = \nabla \cdot z^*(q_h) \tag{5b}$$

where $z^*(q) = \sum_{j \in \mathrm{V}} z_j^*\,(q_j)$ is the conjugate of z.

The proofs of Lemma 1 and Lemma 2 are omitted due to space requirements and can be found in literature [38], [39].

The hypotheses in CE experiments are typically in the form of "the events we are injecting into the system will not cause the system's behavior to change from steady state" [10]. Since the target metric setpoints can be determined via the optimization of Eq. (4), and since according to Lemma 2, u_h is the unique primal optimizer of Eq. (4), then we can conclude that such formulation leads to a close-form solution not causing the system behavior to deviate from the steady state conditions. The close-form solution enables the analytical expression with a number of known functions in order to follow the concept of a well-defined steady behavior and hypotheses, in comparison with a problem solved with a numerical solution which would be only an approximate.

3) Varying ICS process events: In any system, and therefore in any ICS, unpredictable events attributed to hardware or software faults/malfunctions, adverse malicious data, extreme operating conditions (e.g., high temperatures), etc. can lead to certain system conditions in which can further lead to system or sub-systems failures. In CE, the goal is to induce real-world events into the ICS which can be controlled. Following the formulation of the previous CE model, we follow the paradigm of linear distributed control or typically termed distributed averaging proportional integral (DAPI) control, often deployed in multi-agent networked systems in order to control system elements towards a consensus objective [40]. Under this concept, the system components/agents in the distributed ICS (e.g., PLC devices) exchange information with other agents in order to

take the proper control decisions (locally), often using linear PI (proportional integral) controls in an effort to minimize the agents' differences. If we consider the $\mathcal{N}$ agents of the ICS to follow DAPI control, then the control dynamics of $u_j(t)$, $j \in \mathcal{N}$, can be represented as follows:

$$\dot{u}_j(t) = \sum_{\ell \in \mathcal{N}, \ell \neq j} a_{j\ell}\,(x_\ell(t) - x_j(t)) + \beta_j(t) \tag{6}$$

a_{ik} follows Definition 2 and it is a binary index of the unidirectional connectivity: if $a_{ik} = 1$, node i can receive data from k, and if $a_{ik} = 0$ there is no unidirectional communication from k to i. $\beta_j(t)$ can be seen as the bias term to balance ICS operating set-points.

Let C be a diagonal matrix to include the coefficients of the local measurement C_j with $j = \{1, \ldots, n\}$. The differentiation of the local measurement data from other data sent from other system agents (remote) can lead to re-writing Eq. (6) as follows:

$$\dot{u}_j(t) = -C_j x_j(t) + \sum_{\ell \in \mathcal{N}, \ell \neq j} a_{j\ell} x_\ell(t) + \beta_j(t) \tag{7}$$

in which the terms represent the local information, the remote information, and the system bias, respectively. Following Definition 2 of the adjacency matrix representing the remote connectivity index of the ICS network, we can organize Eq. (7) as:

$$\dot{u}(t) = (-C + A)x(t) + \beta(t) = -Qx(t) + \beta(t), \tag{8}$$

where $Q = C - A$. According to [41], and because Q is essentially a positive semi-definite matrix with every row sum being zero, the system has a steady-state equilibrium in consensus based on the control updates of Eq. (7) if the ICS network agents form an undirected and connected graph. When $\beta = 0$, any system equilibrium is in a consensus: by $x_j = x_i, j, i \in \mathcal{N}$.

4) Run experiments in ICS production & experiment automation: A major difference of CE compared to accepted forms of security testing is the automation within the system's production environment as well as the focus on the overall system behavior. Security automation drives continual supervision of the system, which especially in ICS is required due to continuous system changes, e.g., valves pressure, poisoning data, etc. We cannot be aware *a priori* which conditions to the production environment will change the results of a CE test. In this experimentation stage, ideally, the tests need to be tested on the production environment. However, if that is not possible, experiments must be examined in testing conditions as close as possible to the production environment in order to reduce risks in terms of experiments validity increased reliability, and provide enhanced insight into the performance of the experiments in terms of results confidence.

5) Minimize the Blast Radius: CE experiments should contribute in minimizing the blast radius of adverse events to the overall system operation, i.e., limit the range and severity of injected events to the system. A resilient ICS resulted from the CE experiments should reduce to the maximum the

effects of adverse events. Such events, via the continuous CE experimentation, could be fully characterized providing full understanding of the initial conditions that drove them. Thus, it is of paramount importance to develop probabilistic models for asset availability, accessibility, and usability during and succeeding a major event. The task is complex and is dependent on multiple, interacting and often counterintuitive correlations between variables.

IV. USE-CASE SCENARIO

In this section, we introduce a use-case for CE on ICS that realizes the generic CE model presented in Section III on a real-world case study. The work is based on the industrial process control model of the Tennessee Eastman Challenge (TEC) [42]. TEC defines a real-world industrial process control system (IPCS) for continuous chemical manufacturing plants that consists of five units: a reactor, a product condenser, a vapor-liquid separator, a recycle compressor, and a product stripper for producing the desired end products. The model has 12 actuators (manipulated variables) for control and 41 sensors (measured inputs) for monitoring. The model's ICS process produces two products from four input reactants.

The presented use-case utilizes a variation of the IPCS of this testbed as set up by National Institute of Standards and Technology (NIST) to define an example where an adverse event can be detected through CE [43]. This version of the model assumes a decentralized process controller for control loop continuous operation. It includes a PLC with industrial network protocol capability to provide communication between the field equipment and a PLC over a typical protocol (e.g., MODBUS). Field sensors send measured variable information to the PLC, and the PLC uses sensor inputs to compute desired values of the actuators. The IPCS network is segmented from the main network by a boundary router. Routing uses dynamic routing protocols to communicate with the main switch. The testbed utilizes two virtual networks, a main control room network (LAN 1) and the process operation environment (LAN 2). Network traffic between the two subnets is transferred through the boundary router. The operation environment consists of the plant's field sensor simulators, a PLC, and the data historian, while the control room consists of a human–machine interface (HMI) and the controllers. The overall network architecture is depicted in Fig. 3.

A. Step-by-step Description on a Real-world Environment

For the purposes of introducing CE experiments in ICS, we could simplify the TEC testbed to only include one chemical reaction process that outputs *product G* using three input components. We could also restrict potential boundary conditions to the *reactor temperature*, *output yield of product G*, and *input feed rate* of input materials such that the reaction rates of the reactants are a function of the reactor temperature. Product G can be seen as the output of the chemical reaction of three input components: $u_k > 0, k \in \mathbb{N} : k \in [1, 3]$ by sending vector signals $u = [u_1 \dots u_3]^T$ to the chemical process and receiving back sensor measurements $y = [y_1 \dots y_3]^T$. Sample
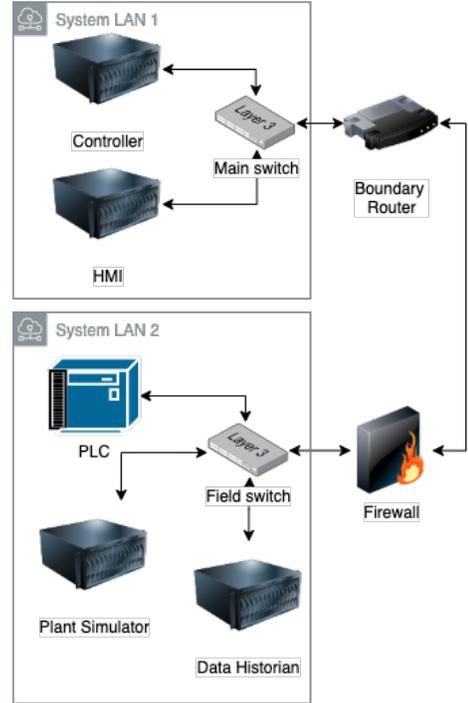

Fig. 3: Network diagram of proposed testbed.

Nominal Input	Description	Value
u_1	Chemical component A feed setpoint to controller	$0.25\ kscmh$
u_2	Chemical component B feed setpoint to controller	$3686\ kg/h$
u_3	Chemical component C feed setpoint to controller	$9.35\ kscmh$

Table I: Sample nominal input values for chemical process production [44].

nominal input values for the chemical process are described in Table I.

1) Steady state definition: In order to setup a CE experiment on the aforementioned use-case, we first need to define the steady state of the system for that chemical process. We can describe this example as a stochastic state space model with additive noise such that it captures the evolution of the chemical process in the plant by describing wide ranges of time constants [45]. Output deviations $\Delta y(t)$ on the production of G and input ranges $u_i(t)$ for states regulation are defined as members of a same set $\mathbb{R}^m$. $\mathcal{V} \subseteq \mathbb{R}^m \times \mathbb{R}^{n_\delta}$ includes nominal input ranges for normal operating limits that induce acceptable states/members of set $\mathcal{W} \subseteq \mathbb{R}^n$. Composition measurements can be omitted to avoid a multi-rate sampling problem [44]. The original publications that introduced the TEC testbed list the plant's process nominal input demands as ranges of input values for equipment protection [42], [43]. Variable value ranges are restricted by existing operating constrains of the TEC's chemical process, within finite domains of operation. These ranges are presented in Table II. Ranges define the normal operating limits as well as the process shutdown limits

	Normal operating limits		Shutdown operating limits	
Process variable	**Low limit**	**High limit**	**Low limit**	**High limit**
Reactor pressure	None	$2895\ kPa$	None	$3000 kPa$
Reactor level	50% ($11.8\ m^3$)	100% ($21.3\ m^3$)	$2.0\ m^3$	$24.0\ m^3$
Reactor temperature	None	$150°C$	None	$175°C$
Product separator level	30% ($3.3\ m^3$)	100% ($9.0\ m^3$)	$1.0\ m^3$	$12.0\ m^3$
Stripper base level	30% ($3.5\ m^3$)	100% ($6.6\ m^3$)	$1.0\ m^3$	$8.0\ m^3$

Table II: Process operating constraints [42].

System Boundary metric	Description
Throughput Rate	Measures amount of end products produced over a specific amount of time.
Input Feed Rate	Measures amount of input materials consumed over a specific amount of time.
Output Yield	Measures percentage of the end product produced at the output of the process.

Table III: Manufacturing process key performance indicators (KPIs) as boundary metrics [43].

for all chemical processes. At each time t, the state $x_j(t)$ of the j_{th} ICS component that participates in the chemical production process lies within the state space $\mathbb{R}^n$, which is represented by the normal and shutdown values of five process variables (e.g., reactor temperature ranges).

A CE experiment on the TEC testbed can utilize operational metrics such as the *throughput rate*, *input feed rate*, or *output yield* of a checmical process to monitor for potential deviations y_t that capture the execution of the chemical process at the system boundary. These metrics, depicted in Table III, can act as primary indicators of the reliability of the industrial process and characterize the hypothesis of the steady state behavior. The operational metrics capture the ICS chemical process's availability at the system boundary, and not technical metrics such as reactor temperature or pressure levels since this is not in line with CE experimentation.

2) Hypothesis formulation: Steady state conditions allow for deviations within the set of normal operating states as depicted in Table II for all process variables of the ICS components. For example, for the aforementioned chemical process of product G, the steady state can be seen as the difference of the dependent process variable values (e.g., the reactor temperature or any operational metric) and the chemical process's control inputs for the production of G.

Following Definition 2, all networked systems along with the plant simulator that participate in this chemical process can be represented in a weighted, directed graph that has at least one globally reachable node (i.e., its left-eigenvector $X_{L_j} \geq 0$, and $X_{L_j} > 0$). We assume agents of the TEC architecture to follow DAPI control with state vectors $x_j(t)$ seen as a linear combination of control inputs $u_j(t)$, $j \in \mathcal{N}$ and outputs $\Delta y(t)$ over time t with a sampling data period of $\Delta t = 1$ min, computed from data since the initial startup of the plant's process.

For the purposes of the CE experiment, we form the hypothesis that the output of the "*output yield*" metric will remain within acceptable bounds, given any set of linear combinations of control inputs [10]. Output can be mapped as a single output y_t vector in $\mathbb{R}^m$. The "*output yield*" metric is chosen as an example. Different types of key performance indicators (KPIs) can be chosen, based on operator knowledge and on the characteristics of the system under test, e.g., manufacturing process performance KPIs, computing resources performance KPIs, OPC data exchange performance KPIs, etc.

3) Varying ICS process events: The CE experiment will vary potential real-world events to affect the modeled steady state and then analyze its impact on the boundary metric "*output yield*" as described in Table III. These variations will mostly take the form of non-critical events that can still affect the overall ICS functionality and disrupt the ICS steady state.

Agents in the used TEC closed-loop system theoretically ensure the reliability of the chemical process during the occurrence of all potential events. Still, when certain limits are crossed in critical and non-critical systems, ICS production and state of operation may not satisfy system nominal demands through the disutility $z(u_h)$ of an ICS component that participates in the chemical process to create product G, as defined in Eq. 4. Furthermore, close-loop processes may often be disrupted and fall back to alternate operations. In such events, ICS often utilize backup analog mechanisms (e.g., meters, alarms, etc.) to continue observing system states. Still, this behavior may not be desirable in the long-term since it may require additional time and effort to perform necessary monitoring or control functions, which may present risks due to reduced quality, safety, or efficiency of the system [46].

The proposed CE experiment focuses only on the ICS process for the production of end product G as captured by the output yield metric over time periods t. The process evolves according to the state vector x_t and inputs u_t. A list of such variations for the presented testbed can be quite diverse, including (but not restricted to) the following:

- Terminate/overload server instances (e.g., historian, HMI).
- Inject latency into requests between the controller and the sensors.
- Inject network errors between the controller and the actuator.
- Make a subnet unavailable (router failure).
- Restart the PLC at a given point in the industrial process.
- Fail a sensor.

This list includes sample input events chosen due to their ability to multiply the effect of the CE experiment and introduce fluctuations at a broad system level (in our case, as a significant deviation on the *output yield rate* for the chemical process of product G) instead of granular, specific technical inconsistencies. We can effectively model these fluctuations as step changes in the chemical process using a simple linear

model [44]. Typical ICS experiments can last from 24 to 48 hours and capture information to calculate relevant boundary metrics over the IPCS potential nominal input values [43]. As depicted in Eq. (9), inputs can be seen as a sequence of signals, where $u_i^0(t)$ is the nominal input for the base operating mode and reference inputs given in Table I. Following a similar case study [44], we can perturb manipulated input variables as three-level sequences (e.g., pseudo random binary sequences (PRBS)) to generate deterministic input signals $u_j(t)$ and reduce the effect of non-linearities on the resulting linear model [47].

$$u_i^0(t) = \begin{cases} u_i^0(t), & t < t_0 \\ u_i^0(t) + \Delta u_i(t), & t_0 \leq t < (t_0 + 24h) \\ u_i^0(t) - \Delta u_i(t), & t_0 + 24h \leq t < (t_0 + 48h) \\ u_i^0(t), & t \geq (t_0 + 48h) \end{cases}$$

$$(9)$$

Example use-case: The PLC scans for signals gathered from the sensors monitoring the reactor. If there is an increase in reactor temperature, then the PLC will communicate with the actuator to initiate commands for decreasing the temperature at the reactor. An adverse event introduces lots of traffic at the protocol level and causes a congestion failure at the boundary router. Field metrics and scans do not reach the control room in timely order, although the system does not yet detect any disconnection to the field. As such, the reactor temperature rises and the output yield of product G decreases to unacceptable levels. From a business process perspective, this event sums up to whether the generator is able to reliably operate on any acceptable steady state and deliver its services in any case scenario. To analyze such boundary system states from a functional perspective, we map the system's operational "output yield" output for product G that reliably sets a vector of measured outputs to monitor for states that do not satisfy system nominal demands.

4) Run experiments in ICS production & analyze results: During CE experimentation, we vary control inputs in an non-binary assertion logic. By following Eq. (5b), the CE experiment aims to measure the divergence (whether positive or negative) from any steady state operation of the system towards a state that does not satisfy system nominal demands.

In the model test-case, unpredictable events on networked components should be used as input to the CE experiment. These events must ideally follow a non-deterministic approach that still conforms to real-world execution conditions of the chemical process for producing product G. For example, an experiment can utilize software or hardware malfunctions, or extreme input to controllers but obviously cannot generate events that can never happen on the real-world (e.g., assume different ICS components or changes in the system's structure). If the CE experiment is run on a real production environment, experimentation must implement safety measures that will enable operators to recover individual ICS components to previously defined steady states [33]. This safety practice in CE experiments is in line with best practices by NIST, which specifically state that exhaustive testing of systems is
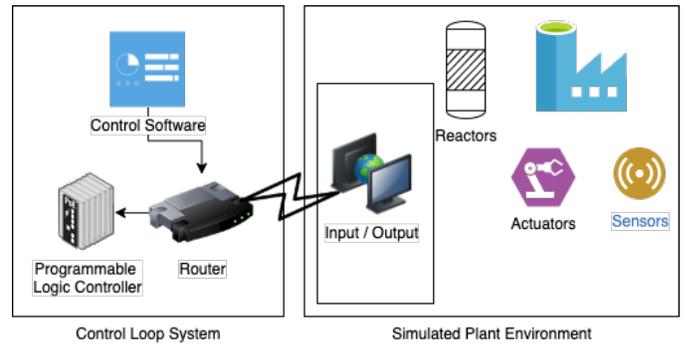

Fig. 4: A hardware-in-the-loop (HIL)-based ICS testbed.

essential to support availability of services [48]. This may include saving and restoring configuration data as a form of ICS component state, or even utilizing in-memory data to mitigate effects in erroneous system states. For all intends and purposes, if semi-simulated environments are used for running CE experiments, then each recorded adverse event should be externally validated to assure operators that all unwanted states detected will generalize to the real system, or is simply a product of experimentation on the testbed that will not reflect to real-world systems [10].

B. Mitigating Operational Risks

Traditionally, CE experiments are purposely run in production to analyze the behavior of the entire real-world system and capture potential third-party influence issues (e.g., inputs, other systems, library dependencies, etc.). Still, the ICS environment is heavily dependent on constant availability, which renders such approaches too risky. Instead, CE events can be simulated using a hardware-in-the-loop (HIL) implementation of the aforementioned model. Such testbeds are efficient when challenging the resiliency of ICS against highly complex scenarios [49], since they integrate the complexity of the real-world system by adding the actual control system and simulating the plant equipment. This way, they mimic real-world systems without field devices and machinery, while utilizing real hardware for the control loop. In HIL testbeds, the software layer must reflect how each added component to the ICS increases the attack surface [3], [50]. An example layout of HIL testbeds in presented in Fig. 4.

CE experiments conducted on HIL testbeds can test actual control systems without having to experiment on real-world field devices and eliminating test risk. We can simulate failure events by selectively failing the aforementioned systems on the HIL testbed and analyze the resulting execution states of engaged hardware devices (e.g., sudden termination of the historian, or take a router offline). Since synthetic models cannot possibly capture the real-world complexity or randomness of events, it is important to utilize as many devices as possible on the HIL testbed. Ideally, only field machinery, actuators and sensors should be simulated, to avoid real-world impact.

V. Conclusions and Future Work

Research has intensively explored systems and software testing, using various approaches such as execution path analysis, injection, fuzzing and conditional detection methods. Still, current work seems inadequate to capture the complexity of functionality, specifications and user behavior in modern distributed and industrial systems. In this paper, we adopt the concept of CE and present how it can be utilized for complex CPS, and specifically, ICS environments in which adverse events can affect the operating system states. We have examined the related work in the area and set up a formal CE model of nonlinear ICS that generalizes over the CE experimentation steps. We have also described a use-case scenario based on the TEC chemical process. In our future work, we aim to develop a resilience assessment framework in a methodological approach that will utilize the concept of CE to quantify the resilience of industrial CPS. The goal is not only to expand the theoretical and experimentation contribution of this work, but also develop automated restoration policies against classes of adverse events which can be utilized in practical systems. Furthermore, we plan to identify necessary and sufficient conditions which can ensure steady state system operation under stealthy cyberattacks and evaluate them in a realistic HIL testbed.

References

[1] P. Verissimo, M. Correia, N. F. Neves, and P. Sousa, "Intrusion-resilient middleware design and validation," *Information Assurance, Security and Privacy Services*, vol. 4, pp. 615–678, 2009.

[2] T. R. McJunkin and C. G. Rieger, "Electricity distribution system resilient control system metrics," in *2017 Resilience Week (RWS)*, 2017, pp. 103–112.

[3] I. Zografopoulos, J. Ospina, X. Liu, and C. Konstantinou, "Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies," *IEEE Access*, vol. 9, pp. 29 775–29 818, 2021.

[4] C. Koulamas and A. Kalogeras, "Cyber-physical systems and digital twins in the industrial internet of things [cyber-physical systems]," *Computer*, vol. 51, no. 11, pp. 95–98, 2018.

[5] F. Yu, Y. Hu, T. Zhang, and Y. Jin, "Resilient distributed estimator with information consensus for cps security," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 41–44.

[6] R. Nateghi, "Multi-dimensional infrastructure resilience modeling: An application to hurricane-prone electric power distribution systems," *IEEE Access*, vol. 6, pp. 13 478–13 489, 2018.

[7] C. Konstantinou, M. Sazos, A. S. Musleh, A. Keliris, A. Al-Durra, and M. Maniatakos, "Gps spoofing effect on phase angle monitoring and control in a real-time digital simulator-based hardware-in-the-loop environment," *IET Cyber-Physical Systems: Theory Applications*, vol. 2, no. 4, pp. 180–187, 2017.

[8] R. Miles, *Learning Chaos engineering: discovering and overcoming system weaknesses through experimentation*. O'Reilly Media, 2019.

[9] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, and C. Rosenthal, "Chaos engineering," *IEEE Software*, vol. 33, no. 3, pp. 35–41, 2016.

[10] C. Rosenthal, L. Hochstein, A. Blohowiak, N. Jones, and A. Basiri, *Chaos Engineering: Building Confidence in System Behavior Through Experiments*. O'Reilly Media, 2017.

[11] B. Beckert, M. Ulbrich, B. Vogel-Heuser, and A. Weigl, "Regression verification for programmable logic controller software," in *International Conference on Formal Engineering Methods*. Springer, 2015, pp. 234–251.

[12] S. Biallas, J. Brauer, and S. Kowalewski, "Arcade. plc: A verification platform for programmable logic controllers," in *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2012, pp. 338–341.

[13] G. Canet, S. Couffin, J.-J. Lesage, A. Petit, and P. Schnoebelen, "Towards the automatic verification of plc programs written in instruction list," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 4. IEEE, 2000, pp. 2449–2454.

[14] J. Nellen, E. Ábrahám, and B. Wolters, "A cegar tool for the reachability analysis of plc-controlled plants using hybrid automata," in *Workshop on Formal Methods Integration*. Springer, 2014, pp. 55–78.

[15] J. Nellen, K. Driessen, M. Neuhäußer, E. Ábrahám, and B. Wolters, "Two cegar-based approaches for the safety verification of plc-controlled plants," *Information Systems Frontiers*, vol. 18, no. 5, pp. 927–952, 2016.

[16] S. Guo, M. Wu, and C. Wang, "Symbolic execution of programmable logic controller code," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 326–336.

[17] M. Zhang, C.-Y. Chen, B.-C. Kao, Y. Qamsane, Y. Shao, Y. Lin, E. Shi, S. Mohan, K. Barton, J. Moyne *et al.*, "Towards automated safety vetting of plc code in real-world plants," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 522–538.

[18] Z. Luo, F. Zuo, Y. Jiang, J. Gao, X. Jiao, and J. Sun, "Polar: Function code aware fuzz testing of ics protocol," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–22, 2019.

[19] M. Muench, J. Stijohann, F. Kargl, A. Francillon, and D. Balzarotti, "What you corrupt is not what you crash: Challenges in fuzzing embedded devices." in *NDSS*, 2018.

[20] D. Tychalas and M. Maniatakos, "Iffset: In-field fuzzing of industrial control systems using system emulation," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 662–665.

[21] N. Neves, J. Antunes, M. Correia, P. Verissimo, and R. Neves, "Using attack injection to discover new vulnerabilities," in *International Conference on Dependable Systems and Networks (DSN'06)*, 2006, pp. 457–466.

[22] J. Antunes, N. Neves, M. Correia, P. Verissimo, and R. Neves, "Vulnerability discovery with attack injection," *IEEE Transactions on Software Engineering*, vol. 36, no. 3, pp. 357–370, 2010.

[23] I. Beschastnikh, Y. Brun, S. Schneider, M. Sloan, and M. D. Ernst, "Leveraging existing instrumentation to automatically infer invariant-constrained models," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 267–277.

[24] T. Ohmann, M. Herzberg, S. Fiss, A. Halbert, M. Palyart, I. Beschastnikh, and Y. Brun, "Behavioral resource-aware model inference," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, 2014, pp. 19–30.

[25] M. R. Aliabadi, A. A. Kamath, J. Gascon-Samson, and K. Pattabiraman, "Artinali: dynamic invariant detection for cyber-physical system security," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 349–361.

[26] Y. Chen, C. M. Poskitt, and J. Sun, "Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 648–660.

[27] A. Basiri, L. Hochstein, N. Jones, and H. Tucker, "Automating chaos experiments in production," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 31–40.

[28] M. A. Chang, B. Tschaen, T. Benson, and L. Vanbever, "Chaos monkey: Increasing sdn reliability through systematic network destruction," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 371–372. [Online]. Available: https://doi.org/10.1145/2785956.2790038

[29] L. Zhang, B. Morin, P. Haller, B. Baudry, and M. Monperrus, "A chaos engineering system for live analysis and falsification of exception-handling in the jvm," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[30] J. Simonsson, L. Zhang, B. Morin, B. Baudry, and M. Monperrus, "Observability and chaos engineering on system calls for containerized applications in docker," 07 2019.

[31] P. Przystałka and W. Moczulski, "Methodology of neural modelling in fault detection with the use of chaos engineering," *Engineering Applications of Artificial Intelligence*, vol. 41, 05 2015.

[32] K. Aihara, "Chaos engineering and its application to parallel distributed processing with chaotic neural networks," *Proceedings of the IEEE*, vol. 90, no. 5, pp. 919–930, 2002.

[33] K. A. Torkura, M. I. H. Sukmana, F. Cheng, and C. Meinel, "Security chaos engineering for cloud services: Work in progress," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 2019, pp. 1–3.

[34] ——, "Cloudstrike: Chaos engineering for security and resiliency in cloud infrastructure," *IEEE Access*, vol. 8, pp. 123 044–123 060, 2020.

[35] K. Paridari, N. O'Mahony, A. El-Din Mady, R. Chabukswar, M. Boubekeur, and H. Sandberg, "A framework for attack-resilient industrial control systems: Attack detection and controller reconfiguration," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 113–128, 2018.

[36] O. Analytica, "Us cyberattack underlines sub-national risks," *Emerald Expert Briefings*, no. oxan-es.

[37] V. N. Phat and P. T. Nam, "Exponential stability and stabilization of uncertain linear time-varying systems using parameter dependent lyapunov function," *International Journal of Control*, vol. 80, no. 8, pp. 1333–1341, 2007.

[38] J. Schiffer and F. Dörfler, "On stability of a distributed averaging pi frequency and active power controlled differential-algebraic power system model," in *2016 European Control Conference (ECC)*, 2016, pp. 1487–1492.

[39] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[40] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[41] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[42] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Computers & chemical engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[43] C. Tang and C. Tang, *Key performance indicators for process control system cybersecurity performance analysis*. US Department of Commerce, National Institute of Standards and Technology, 2017.

[44] B. C. Juricek, D. E. Seborg, and W. E. Larimore, "Identification of the tennessee eastman challenge process with subspace methods," *Control Engineering Practice*, vol. 9, no. 12, pp. 1337–1351, 2001.

[45] V. Krishnamurthy, *Partially observed Markov decision processes*. Cambridge university press, 2016.

[46] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security," *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

[47] K. Godfrey, *Perturbation signals for system identification*. Prentice Hall International (UK) Ltd., 1993.

[48] A. A. Jillepalli, F. T. Sheldon, D. C. de Leon, M. Haney, and R. K. Abercrombie, "Security management of cyber physical control systems using nist sp 800-82r2," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1864–1870.

[49] W. Grega, "Hardware-in-the-loop simulation and its application in control education," in *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No. 99CH37011*, vol. 2. IEEE, 1999, pp. 12B6–7.

[50] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A.-R. Sadeghi, M. Maniatakos, and R. Karri, "The cybersecurity landscape in industrial control systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1039–1057, 2016.