Data-driven Vulnerability Analysis of Networked Pipeline System

Yu Zheng[1] *GSIEEE*, Olugbenga Moses Anubi[1] *SMIEEE*

Abstract— **This paper introduces an attack generation framework for evaluating the vulnerability of nonlinear networked pipeline systems. The vulnerability analysis is formulated as determining the presence of feasible attack sets, defined by boundary functions representing the effectiveness and stealthiness of attack signals with respect to the objective and attack detection module. The framework utilizes three data-driven models, including two discriminative models that learn the boundary functions and a generative model that produces elements of the feasible attack set. A new loss function ensures successful attack generation with high probability.**

I. INTRODUCTION

Networked pipeline systems (NPS) play a crucial role in transporting essential resources such as oil and gas. With the increasing integration of information and control techniques in these systems, they are becoming more vulnerable to cyber attacks, which can have severe consequences on the safety, security, and reliability of pipeline operations. Examples include the recent Colonial pipeline ransomware attack [1], Stuxnet attack [2], and more. Supervisory control and data acquisition (SCADA) is a critical part of NPS containing data collection, state estimation, and decision processes. The state estimation process could be misled maliciously by compromising only a small portion of the IoT-based measurement system [3]. Modifying the control inputs at the automatic control layer could result in catastrophic consequences on physical assets [4].

Assessing the vulnerability of NPS is a challenging task, as it requires a thorough understanding of the complex interplay between the physical and cyber components of the system and the potential attack scenarios. Several attack generation techniques have been proposed in the literature to address the vulnerability analysis problem in cyber-physical systems (CPS). These techniques range from mathematical and analytical methods to data-driven and simulation-based approaches. For example, the authors [5] define vulnerability under sensor attacks as the boundedness of the estimation error, and derive sufficient and necessary conditions through analysis of the system's reachable unstable zero dynamics. The authors in [6] characterize attacks using an asymptotic detection performance (ADP), defined as the rate of decrease in the worst-case probability of error. However, the nonlinear nature of many networked pipeline systems makes the attack generation problem challenging. Recent approaches have begun to explore sample-based solutions. Some works have trained generative adversarial networks (GANs) to learn from existing attack datasets [7], but this relies on the existence of non-generative attack datasets and good representative quality of the training data. The authors in [8] used data-driven models to approximate the system model, which reduces the complexity of the attack generation problem.

[1]Yu Zheng and Olugbenga Moses Anubi are with the Department of Electrical and Computer Engineering, Florida State University, FL, USA. `yzheng6@fsu.edu, oanubi@fsu.edu`

While these methods have been shown to be effective in various applications, they are limited in their ability to either capture the nonlinear nature of networked pipeline systems or the lack of prior non-generative attack datasets. In this paper, we present an attack generative framework that addresses these limitations by integrating physical runtime data of the pipeline system and a data-driven attack generation approach. This approach can be applied to both linear and nonlinear systems without relying on the system's model information or prior attack datasets.

The remainder of the paper is organized as follows. In section II, the notations used in the paper are given. In section III, we present the models of the networked pipeline system and formulate the attack generation problem. In section III, we present the data-driven vulnerability analysis framework, introduce a new loss function, and propose a training procedure to avoid biased learning. In section V, we implement the proposed method on a case study of a gas pipeline segment. All conclusive remarks follow in section VI.

II. NOTATIONS

We use $\mathbb{R}^n, \mathbb{R}_+$ to denote the space of real vectors of length n and positive real numbers respectively. The probability triple is denoted by $(\Omega, \mathcal{F}, \mathbf{P_z})$ [9], where Ω is a sample space containing the set of all possible outcomes, $\mathcal{F}$ is an event space, and $\mathbf{P}_z$ is the associated probability functions of the events in $\mathcal{F}$. We use the symbol $\mathbb{E}$ to denote the expected value operator. An undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ contains vertices, denoted by $\mathcal{E} = \{v_1, v_2, \dots, v_n\}$, and edges, denoted by $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$. (v_1, v_2) and $(v_1, v_2) \in \mathcal{E}$ represents an edge in $\mathcal{G}$. Consequently, the adjacency matrix, denoted by $A(\mathcal{G})$, is a square matrix of size $|\mathcal{V}|$, defined as [10]

$$A_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{otherwise} \end{cases} .$$

III. MODEL DEVELOPMENT

In this section, we introduce the models of the networked pipeline system, including the physical plants, network dynamics, a supervisory control and data acquisition (SCADA) system, and an attack model. Fig 1 illustrates the system, featuring two control layers. The supervised control layer collects measurement data from IoT sensors across the network, estimates system states, identifies measurement errors,

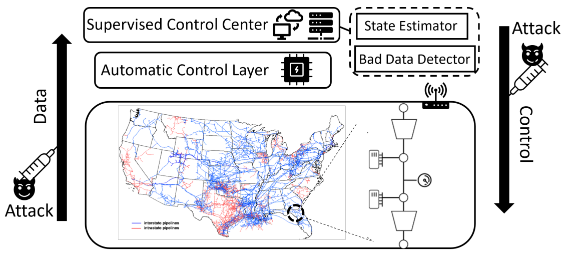

Fig. 1. A schematic diagram of SCADA for networked pipeline system under adversarial attacks

and generates the reference for the local controllers. The automatic control layer consists of lower-level controllers of the compressor stations.

A. System Model

A nonlinear creep flow model is used to describe the gas flow in a pipeline segment [11]:

$$\frac{\partial p}{\partial t} = -\frac{c^2\rho}{G}\cdot\frac{\partial Q_n}{\partial x}, \quad \frac{\partial p}{\partial x} = -\frac{2f\rho^2c^2Q_n^2}{DG^2p}, \tag{1}$$

where p is the pressure in the pipeline, x is the position of flow, c is the speed of sound in gas $[\mathsf{m/s}]$, ρ is the average gas density over the cross-section area of the pipeline $[\mathsf{kg/m}^3]$, Q_n is the volumetric flow at standard conditions, f is the friction factor, D is the diameter of the pipeline and $G = \pi(D/2)^2$. While the partial differential equation (PDE) model provides an accurate description of the gas flow dynamics for infrequent online optimization, an ordinary differential equation (ODE) model is sufficient to describe the pressure dynamics at nodes [11]. The dynamical pressure model at node i is given as

$$\dot{p}_i = \frac{c^2}{V_i}\sum_{j\in\mathcal{N}_i}\sqrt{\frac{|p_j^2-p_i^2|}{K_{ij}}}\mathsf{sgn}(p_j-p_i) - w_i, \tag{2}$$

where $K_{ij} = \frac{64f_{ij}c^2\Delta x_{ij}}{\pi^2D_{ij}^5}$, $V_i = \frac{\pi}{8}\sum_{j\in\mathcal{N}_i}D_{ij}^2\Delta x_{ij}$, and w_i is the mass flow at pipeline i, Δx_{ij} denotes the length of pipeline between node i and node j, $\mathcal{N}_i$ denote the set of neighborhood pipelines of the pipeline i. Let

$$\psi(p_i,p_j) \triangleq \sqrt{\frac{|p_j^2-p_i^2|}{K_{ij}}}\mathsf{sgn}(p_j-p_i),$$

and

$$\Psi(\mathbf{p}) \triangleq \begin{bmatrix} 0 & \psi(p_1,p_2) & \dots & \psi(p_1,p_n) \\ \psi(p_2,p_1) & 0 & \dots & \psi(p_2,p_n) \\ \vdots & \vdots & \ddots & \vdots \\ \psi(p_n,p_1) & \psi(p_n,p_2) & \dots & 0 \end{bmatrix},$$

then the model of the entire pipeline network is given by

$$\begin{aligned}\dot{\mathbf{p}} &= c^2V^{-1}(A\odot\Psi(\mathbf{p}))\mathbf{1} - \mathbf{w} \\ &= c^2V^{-1}(A\odot\Psi(\mathbf{p}))\mathbf{1} + B_{\text{sup}}\mathbf{w}_{\text{sup}} - B_{\text{dem}}\mathbf{w}_{\text{dem}} \\ &\triangleq f(\mathbf{p},\mathbf{w}),\end{aligned} \tag{3}$$

where $V^{-1} = \mathsf{diag}([V_1^{-1}, V_2^{-1}, \dots, V_n^{-1}])$, $\mathbf{w}_{\text{sup}}$ is a vector of mass inflows at the supply points and $\mathbf{w}_{\text{dem}}$ is a vector of mass outflows at the demand points. Clearly $\mathbf{w} = -B_{\text{sup}}\mathbf{w}_{\text{sup}} + B_{\text{dem}}\mathbf{w}_{\text{dem}}$, where B_{dem} and B_{sup} are appropriately dimensioned matrices such that $B_{\text{sup}}^\top B_{\text{dem}} = 0$ and $P\left[\, -B_{\text{sup}} \mid B_{\text{dem}} \,\right] = I$ for some projection matrix P. In addition, a dynamical model of the compressor between two nodes i and j is given by [12]:

$$\begin{aligned}\dot{\omega}_{ij} &= \frac{1}{J_{0_i}}(u_{ij} + \rho r_2^2 q_{ij}\omega_{ij}) \\ \dot{q}_{ij} &= \frac{A_c}{L_c}\left(\phi\left(\omega_{ij},q_{ij}\right)p_i - p_j\right) \\ \dot{w}_{ij} &= q_{ij},\end{aligned} \tag{4}$$

where

$$\phi(\omega,q) = \left(1 + \frac{1}{T_{in}c_p1000}\left(\rho r_2^2\omega^2 + k_fq^2 - \frac{r_1^2}{2}\left(\omega - \frac{q}{A_cr_1\rho_{in}}\right)^2\right)\right)^{\frac{\gamma}{\gamma-1}},$$

ω_{ij} is the compressor rotor angular velocity $[\mathsf{rad/s}]$, q_{ij} is the mass power flow $[\mathsf{kg/s}]$, $\phi(\omega,q)$ is the compressor characterization, u_{ij} is the mechanical torque $[\mathsf{N\text{-}m}]$ applied to the rotor shaft or inertia $J_{0_i}[\mathsf{kg\text{-}m}^2]$. p_i and p_j are the input and output pressures of the compressor, respectively. For the system considered, the compressors are at the extremes of the network. Thus, at each compressor node, one of p_i and p_j will be the ambient pressure. $p_i = P_{\text{amb}}$ for upstream compressors, while $p_j = P_{\text{amb}}$ for downstream compressors.

B. Control Model

A supervised control framework is employed to stabilize pipeline pressure and regulate mass flow, compromising a supervised control layer and an automatic control layer. The supervised control layer contains an estimator, a bad data detector (BDD), and a supervised controller. The pipeline system is equipped with hundreds of smart pressure and flow meters, the goal of the estimator at the supervised control layer is to fuse the meter readings and estimate the states of interest $\mathbf{p}$. The measurement model is given as

$$\mathbf{y} = g(\mathbf{p},\mathbf{w}). \tag{5}$$

Then an unscented Kalman Filter (UKF) is utilized to perform sensor fusion and state estimation. Firstly, following standard unscented transformation, we use $2n+1$ sigma points to approximate the state $\mathbf{p}$ with assumed mean $\bar{\mathbf{p}}$ and covariance $P_{\mathbf{p}}$ as follows:

$$\begin{aligned}&\chi_0 = \bar{\mathbf{p}}, \quad \chi_i = \bar{\mathbf{p}} + (\sqrt{(\lambda+n)P_{\mathbf{p}}})_i, \quad i = 1,\cdots,n, \\ &\chi_{i+n} = \bar{\mathbf{p}} + (\sqrt{(\lambda+n)P_{\mathbf{p}}})_{i-n}, \quad i = n+1,\cdots,2n.\end{aligned}$$

The corresponding weights for the sigma points are given as $W_0^m = \lambda/(n+\lambda)$, $W_0^c = W_0^m + (1-\alpha^2+\beta)$, $W_i = 1/2(L+\lambda)$, and $\lambda = \alpha^2(n+\kappa) - n$ represents how far the sigma points are away from the state, $\kappa \geq 0, \alpha \in (0,1]$, and $\beta = 2$ is the optimal choice for Gaussian distribution. Assume

$\mathbf{p}_{k-1} \sim \mathcal{N}(\bar{\mathbf{p}}_{k-1}, P_{\mathbf{p},k-1})$, the prediction step is given by

$$\hat{\mathbf{p}}_k^- = \sum_{i=0}^{2n} W_i g(\mathcal{X}_{k-1}, \mathbf{w}_k), \;\; \hat{\mathbf{y}}_k^- = \sum_{i=0}^{2n} W_i \mathcal{Y}_{k,i},$$
$$\hat{P}_{\mathbf{p},k} = \sum_{i=0}^{2n} W_i (g(\mathcal{X}_{k-1}, \mathbf{w}_k) - \hat{\mathbf{x}}_k)(g(\mathcal{X}_{k-1}, \mathbf{w}_k) - \hat{\mathbf{x}}_k)^T + R,$$

where $\mathcal{Y}_k = f(g(\mathcal{X}_{k-1}, \mathbf{w}_k))$. The correction step is given by

$$\hat{\mathbf{p}}_k = \hat{\mathbf{p}}_k^- + \mathbf{K}_k(\mathbf{y}_{k,} - \hat{\mathbf{y}}_{k,}), \;\; P_{\mathbf{p},k} = \hat{P}_{\mathbf{p},k} - \mathbf{K}_k \hat{P}_{\mathbf{y},k} \mathbf{K}_k^T, \tag{6}$$

where the Kalman gain is $\mathbf{K}_k = \hat{P}_{\mathbf{py}} \hat{P}_{\mathbf{y},k}^{-1}$ with

$$\hat{P}_{\mathbf{y},k} = \sum_{i=0}^{2n} W_i (\mathcal{Y}_{k,i} - \hat{\mathbf{y}}_{k,})(\mathcal{Y}_{k,i} - \hat{\mathbf{y}}_{k,})^T + Q,$$
$$\hat{P}_{\mathbf{py}} = \sum_{i=0}^{2n} W_i (\mathcal{X}_{k,i}^\star - \hat{\mathbf{p}}_k)(\mathcal{Y}_{(k,i),} - \hat{\mathbf{y}}_{k,})^T,$$

and Q and R are the measurement and process noise covariance matrices respectively. Bad data detector (BDD) is defined as a function $D: \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}_+$ mapping from the state estimates to a detection residual (i.e. 1-norm or 2-norm residual-based detectors [13], [14]) or a detection likelihood (i.e. χ^2 detector [15], [8]).

$$\mathbf{r}_i = D(\hat{\mathbf{p}}_i, \mathbf{y}_i, \mathbf{w}_i). \tag{7}$$

Linearizing the models (3) around the equilibrium point $(\mathbf{p}_{eq}, \mathbf{w}_{eq})$ and discretizing with a fixed time step T_s yields

$$\Delta\mathbf{p}_{k+1} = A\Delta\mathbf{p}_k + B\Delta\mathbf{w}_k, \tag{8}$$

where $\Delta\mathbf{p}_k = \mathbf{p}_k - \mathbf{p}_{eq}$, $\Delta\mathbf{w}_k = \mathbf{w}_k - \mathbf{w}_{eq}$, and

$$A = \left.\frac{\partial f}{\partial \mathbf{p}}\right|_{(\mathbf{p}_{eq}, \mathbf{w}_{eq})} T_s + I_n, \;\; B = \left.\frac{\partial f}{\partial \mathbf{w}}\right|_{(\mathbf{p}_{eq}, \mathbf{w}_{eq})} T_s, \;\; C = \left.\frac{\partial g}{\partial \mathbf{p}}\right|_{\mathbf{p}_{eq}}.$$

Then, a model predictive controller of horizon length h is utilized to generate the reference mass flow $\mathbf{w}^r$:

$$\begin{aligned} \underset{\Delta\mathbf{p}_{[k,k+h]}, \Delta\mathbf{w}_{[k,k+h]}}{\text{Minimize}} &: \sum_{t=0}^{h} \frac{1}{2} \Delta\mathbf{p}_t^\top M_1 \Delta\mathbf{p}_t + \sum_{t=0}^{h-1} \frac{1}{2} \Delta\mathbf{w}_t^\top M_2 \Delta\mathbf{w}_t \\ \text{Subject to} &: \Delta\mathbf{p}_{t+1} = A\Delta\mathbf{p}_t + B\Delta\mathbf{w}_t, \; t = 0, \cdots, h-1 \\ & \quad \Delta\mathbf{p}_0 = \Delta\mathbf{p}_k, \;\; \Delta\mathbf{p}_h = 0. \end{aligned} \tag{9}$$

Thus, the reference mass flow is given as $\mathbf{w}_k^r = \Delta\mathbf{w}_{[k]}^\star + \mathbf{w}_{eq}$. Next, at the automatic control layer, a PID controller is used to control the compressors to track the reference mass flow.

$$u_{ij} = K_P \, e(t) + K_I \int_0^t e(\tau)\mathrm{d}\tau + K_D \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \tag{10}$$

where $e(t) = \mathbf{w}_{ij} - \mathbf{w}_{ij}^r$, and the PID gains K_P, K_I, K_D are designed to achieve $\lim_{t\to\infty} \|e(t)\| = 0$.

C. Attack Model

The attacks could be injected through the sensing process and actuation process, and it can be modeled as [15], [14]

$$\begin{aligned} \dot{\mathbf{p}} &= f(\mathbf{p}, \mathbf{w} + \mathbf{e}^u), \\ \mathbf{y} &= g(\mathbf{p}, \mathbf{w}) + \mathbf{e}^y. \end{aligned} \tag{11}$$

To characterize the effect of attacks in the system (11), effectiveness and stealthiness are two common criteria [3]. We use $l_1 : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}, l_2 : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ to denote functions evaluating the effectiveness and stealthiness of the attacks respectively. Then a feasible set of attacks $\mathcal{S}$ is defined with given thresholds of effectiveness and stealthiness τ_E, τ_S:

$$\mathcal{S} = \left\{ \mathbf{e}^u \in \Sigma_{k_u}, \mathbf{e}^y \in \Sigma_{k_y} \middle| l_1(\mathbf{e}^u, \mathbf{e}^y) \geq \tau_E, l_2(\mathbf{e}^u, \mathbf{e}^y) \leq \tau_S \right\}. \tag{12}$$

Most research uses the estimation error to represent the effectiveness of sensor attacks such that $l_1 = \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|$ [13], [14]. The stealthiness function often employs the BDD function $l_2 = D$, where D is defined in (7).

Consequently, the attack generation problem is finding a generative model of the form

$$G(\mathbf{z}, \theta) : (\Omega, \mathcal{F}, P_\mathbf{z}) \times \mathbb{R}^{n_g} \to \mathbb{R}^n \tag{13}$$

with a constant tunable parameter vector $\theta \in \mathbb{R}^{n_g}$, and a prior probability sample space $(\Omega, \mathcal{F}, P_\mathbf{z})$ with random variables $\mathbf{z} \sim P_\mathbf{z}$, such that

$$Pr\{G(\mathbf{z}, \theta) \in \mathcal{S}\} \geq \alpha \tag{14}$$

for some $\alpha \in (0, 1)$. Essentially, $G(\mathbf{z}, \theta)$ is a generative model for the set $\mathcal{S}$ if $G(\mathbf{z}, \theta) \in \mathcal{S}$ with a high probability given by the lower bound α.

However, obtaining the close-form expressions of l_1 and l_2 is challenging due to their complex composition of the models from (2) to (10). Moreover, searching for feasible attacks in $\mathcal{S}$ is at least an *NP*-hard problem due to the nonlinearity and nonconvexity of the problem. To address these challenges, we proposed a data-driven approach that consists of two discriminators learning l_1 and l_2 from runtime data, and a generator learning how to solve this *NP*-hard attack generation problem.

IV. MAIN RESULTS

In this section, we introduce a solution to search for feasible attacks in the set $\mathcal{S}$ defined in (12) using only the runtime data of NPS. The data-driven attack generation framework is illustrated in Fig. 2. The runtime data is obtained from the physical experiment and is used to guide the training of the discriminators. The trained discriminators then supervise the learning process of the generator.

A. Generator

The generator $G(\mathbf{z}, \theta)$ is a deep neural network learning the distribution of feasible attacks from sampling noise. It is trained with the loss function

$$L(G(\mathbf{z}; \theta)) = \text{ReLU}\left(\underset{\mathbf{z} \sim \mathbf{P}_z}{\mathbb{E}} [D(G(\mathbf{z}; \theta))] - (1 - \alpha) \right). \tag{15}$$

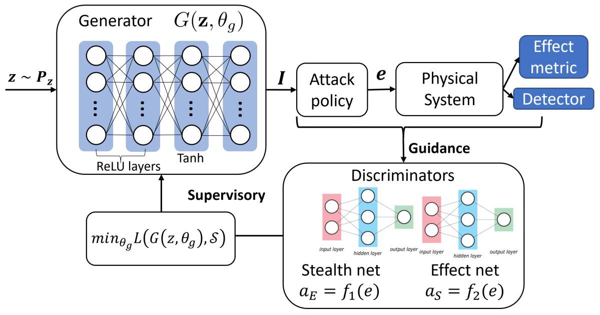

Fig. 2. Schematic description of the proposed vulnerability analysis model

where D function is given by

$$D(\mathbf{e}) = exp[\mathsf{LeakyReLU}_s(\tau_E - l_1(\mathbf{e})) \\ + \mathsf{LeakyReLU}_s(l_2(\mathbf{e}) - \tau_S)], \tag{16}$$

where $\mathbf{e} = \begin{bmatrix} \mathbf{e}^{u\top} & \mathbf{e}^{y\top} \end{bmatrix}^\top$. Then the following minimization program is employed to train the generator and solved by gradient descent.

$$\theta^\star = \underset{\theta}{\mathsf{argmin}} \underset{\mathbf{z} \sim \mathbf{P}_z}{\mathbb{E}} L(G(\mathbf{z}; \theta)). \tag{17}$$

B. Discriminators

Successful training of the generator requires knowledge of l_1 and l_2 functions. However, it is hard to obtain their closed-form expression. Data-driven approximation offers a practical solution, as it only requires runtime data instead of a high-fidelity model and provides easy calculation of gradients. In this paper, we use two deep regression neural networks to approximate l_1 and l_2:

$$a_E = f_1(\mathbf{e}; \theta_E), \quad a_S = f_2(\mathbf{e}; \theta_S). \tag{18}$$

They are both trained with the mean-square-error (MSE) loss function given N runtime effectiveness metric data $l_1(\mathbf{e})$ with the corresponding injection of attacks $\mathbf{e}$:

$$\underset{\theta_E}{\text{Minimize:}} \frac{1}{N} \sum_{j=1}^{N} \left\| l_1(\mathbf{e})^{(j)} - f_1(\mathbf{e}; \theta_E)^{(j)} \right\|_2^2, \tag{19}$$

and given N runtime stealthiness metric data $l_2(\mathbf{e})$ with the corresponding injection of attacks $\mathbf{e}$:

$$\underset{\theta_S}{\text{Minimize:}} \frac{1}{N} \sum_{j=1}^{N} \left\| l_2(\mathbf{e})^{(j)} - f_2(\mathbf{e}; \theta_S)^{(j)} \right\|_2^2, \tag{20}$$

C. Training Algorithm

The generator maps random samples $\mathbf{z} \sim \mathbf{P}_z$ to the parameter vector $\mathbf{I} \in \mathbb{R}^p$ of a pre-defined attack policy $\pi(\mathbf{I})$. Given $\mathbf{I} \in \mathbb{R}^p$, the attack policy $\pi(\mathbf{I})$ is a deterministic time sequence of the injected attacks given by

$$\pi(\mathbf{I}) \triangleq \{t_0(\mathbf{I}), T(\mathbf{I}), g(\mathbf{I})\}.$$

It comprises the start time of the attack injection $t_0(I)$, the duration of the attack injection $T(I)$, and the attack profile $g(\mathbf{I}) : [t_0(\mathbf{I}) \;\; t_0(\mathbf{I}) + T(\mathbf{I})] \to \mathbb{R}^n$. The specific attack policy is assumed to be predetermined for the purpose of this paper.

Biased learning is a concern in deep generative models [16], particularly when trained with multiple interrelated data-driven models. The discriminators might learn biased l_1 and l_2 functions based on a limited space of attacks covered by the generative model. This in turn affects the training of the generator network. To tackle biased learning, this paper incorporates random and generated attack datasets into the discriminator training process. Algorithm 1 outlines the complete training procedure.

Algorithm 1 Training of the attack generative model

```
Hyperparameters: $\tau_E$, $\tau_S$, $\alpha$, $\pi$, $s$
  1) Generate random parameter vector $\mathbf{I}$, implement the attack policy $\pi(\mathbf{I})$ in system experiment and obtain a random attack dataset $\{[I_1, I_2, I_3], [a_E, a_S]\}_{rand}$;
For $i$ in $N_{epoch}$ do:
  2) Pass a batch of $m$ samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(m)}\} \sim \mathbf{P}_z$ through the generator to obtain $[I_1, I_2, I_3] = G(z, \theta_{i-1})$;
  3) Perform corresponding attack policy $\pi(G(z, \theta_{i-1}))$ in system experiment and obtain a generated attack dataset $\{[I_1, I_2, I_3], [a_E, a_S]\}_{gen}$;
  4) Train discriminators using both random attack dataset and the current generated attack dataset $\to$ (19), (20);
  5) Train the generator using the trained discriminators in the loss function (15), where $l_1(\mathbf{e}) = f_1(\mathbf{e}, \theta_{E_i})$ and $l_2(\mathbf{e}) = f_2(\mathbf{e}, \theta_{S_i})$.
End For
```

V. SIMULATION

In this section, we evaluate the proposed attack generative model on a 4-node pipeline system, as depicted in Fig. 3. The objective is to regulate the node pressures at their corresponding equilibrium point $\mathbf{p}_{eq}$, while meeting the demanding mass flow w_{dem} at node 4 and subject to the flow supply w_{h1} and w_{h2} at the two wellheads. To this end, the supervised MPC controller (9) generates the reference mass flow w_1 at node 1 using the UKF estimation (6) of the node pressure and the mass flow in the other pipelines. Subsequently, the local PID controller (10) controls the compressor to regulate the pipeline flow supplied to the node 1.
The topology of the network is represented by the adjacent matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{21}$$

The pipeline model parameters are chosen as $c = 330m/s$, $D_{12} = 0.8m$, $D_{23} = 0.5m$, $D_{34} = 1.5m$, $\Delta x_{12} = \Delta x_{23} = \Delta x_{34} = 10m$, $f_{12} = f_{23} = f_{34} = 0.0025$. The compressor model parameters are chosen as $J_0 = 47.7465$, $\rho r_2^2 = 0.0951$, $T_{in} c_p = 13322.3$, $k_f = 0.05$, $r_1 = 20$, $A_c r_1 \rho_{in} = 0.5834$, $\gamma = 1.2$, and $A_c / r_c = 0.0146$.

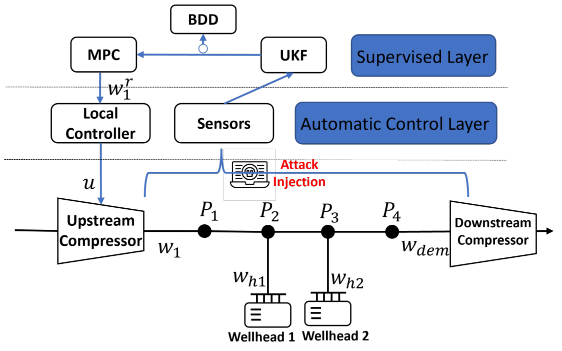

Fig. 3. A 4-node pipeline network as the simulation platform

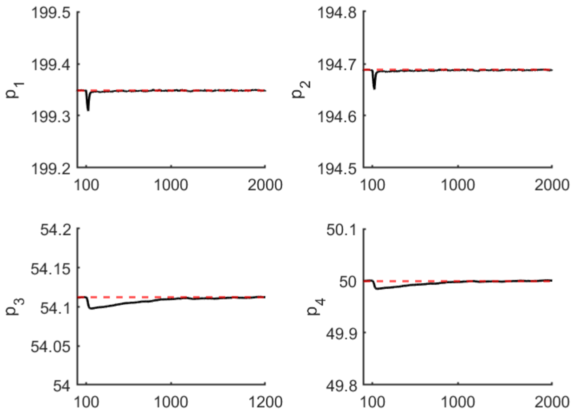

Fig. 4. Nominal control performance with disturbance injected between 100s and 120s (nominal controller has robustness against small disturbance)

In this simulation, we target the equilibrium point $\mathbf{p}_{eq} = \begin{bmatrix}199 & 194 & 54 & 50\end{bmatrix}^\top$, corresponding to the equilibrium point of the mass flow $\mathbf{w}_{eq} = \begin{bmatrix}-20 & -5 & -5 & 13\end{bmatrix}^\top$. Fig. 4 illustrates an example of the nominal control performance, where a disturbance $\mathbf{d} = 0.1$ is injected into pressure measurements during $[100s, 120s]$. The supervisory control framework successfully regulates the node pressures back to their desired equilibrium point. The UKF is used to estimate 4 node pressure $\mathbf{p}_1 \sim \mathbf{p}_4$ and 3 mass flow w_{h1}, w_{h2}, w_{dem} from their noisy measurements. The BDD is then defined as

$$r = \|\mathbf{y} - g(\hat{\mathbf{x}})\|_2, \tag{22}$$

where $\hat{\mathbf{x}} = \begin{bmatrix}\hat{\mathbf{p}}^\top & w_{h1} & w_{h2} & w_{dem}\end{bmatrix}^\top$. This BDD residual r is used for the stealthiness metric. And the effectiveness metric is given by the control error

$$e = \|\mathbf{p} - \mathbf{p}_{eq}\|_2. \tag{23}$$

The objective of the attack generation task is to fool UKF into giving biased estimates $\hat{\mathbf{x}}^a$ leading to small r but big e. The feasible attack set is given particularly as

$$\mathcal{S} = \{\mathbf{e}^y | e \geq 10, r \leq 0.2\}, \tag{24}$$

where $\mathbf{e}^y$ is injected through the measurements as shown in (11). We used an example attack policy called *ramp* attack

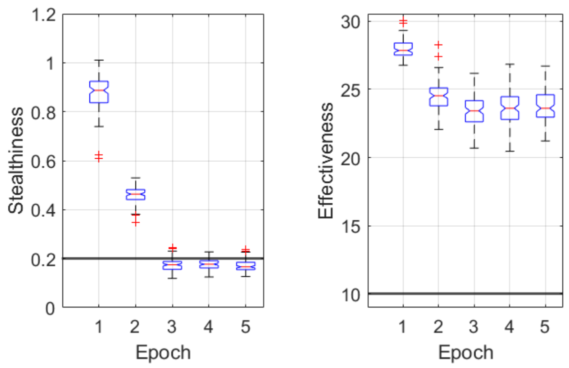

Fig. 5. Testing performance of the attack generator in system simulation at each training epoch (100 sampling test)

policy $\pi(\mathbf{I}) = \{t_0(\mathbf{I}), T(\mathbf{I}), g(\mathbf{I})\}$ through each channel, given by

$$\begin{aligned} t_0 &= t_l + (t_r - t_l)\mathbf{I}_2, \\ T &= T_l + (T_r - T_l)\mathbf{I}_3, \\ e = g(\mathbf{I}) &= \begin{cases} 0 & t < t_0, \\ \min\left(\frac{(t-t_0)}{T}, 1\right)\bar{e}\mathbf{I}_1 & t \geq t_0, \end{cases} \end{aligned} \tag{25}$$

Here $\bar{e}$ is the maximum magnitude of the attack signal. The attack injection start time $t_0 \in [t_l \;\; t_r]$ and the duration time $T \in [T_l \;\; T_r]$. We use $\bar{e} = 0.5$, $t_l = 20s$, $t_r = 40s$, $T_l = 0s$, $T_r = 100s$ and the simulation time for generator training is set as $200s$. These exact values of t_0, T, e at runtime are specified by the parameters $\mathbf{I} \in \mathbb{R}^3$ generated by the generator.

Next, we implemented Algorithm 1 with $\tau_E = 10, \tau_S = 0.2, \alpha = 0.8, s = 0.01$ and π given by (25). The generator employed is a deep neural network composed of 4 layers, *ReLU*, *ReLU*, *Tanh*, *Sigmoid*, with $500, 1000, 500, 3n$ neurons respectively, where n is the number of nodes under attack. The input size is set as 10. The stealth net consists of 3 *ReLU* hidden layers and *Sigmoid* output layer, and the effect net consists of 3 *ReLU* hidden layers and *Linear* output layer. The numbers of neurons at layers are $500, 1000, 500, 1$ for both effect net and stealth net. They are trained by (20) and (19) respectively. Adams optimizer, in Matlab deep learning toolbox, is used with the learning rate 0.0002, gradient decay factor 0.5, and square gradient decay factor 0.999. We train the framework 5 epochs. In each epoch, the generator is trained 10 batches of 5000 batch size, while the discriminators are trained 200 batches of 1000 batch size.

After each epoch of training, the performance of the attack generator is tested in a real system simulation, as shown in Fig. 5. The generated attacks become more stealthy (stealthiness metric value decrease) as the training epoch increases, while their effectiveness is also sacrificed in the first 2 epochs. However, once the stealthiness falls below the threshold 0.2, the generator increases effectiveness from epoch 3 to epoch 5. The F_1 score of generating feasible attacks in the set $\mathcal{S}$ defined in (24) is increasing from 0 to 0.86. The training loss curves of discriminators are shown in Fig. 6. It is seen that, at the first 2 epochs, the training of discriminators does not converge yet, so the performance of the attack generator also has bigger deviations as shown in Fig. 5. As the number of epochs increases, the convergence of the generator and two discriminators arrive simultaneously.

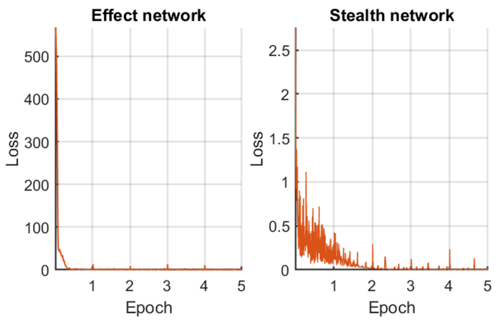

Fig. 6. Loss curves of two discriminators at each training epoch

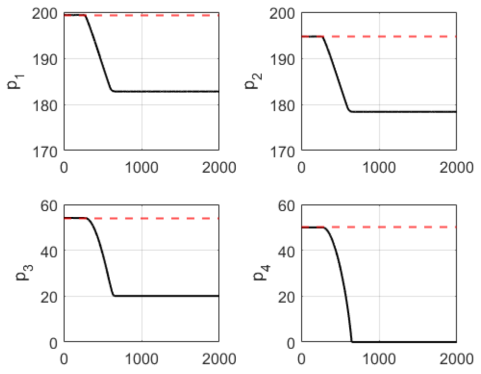

Fig. 7. An example control performance under the generated attacks from the trained attack generator

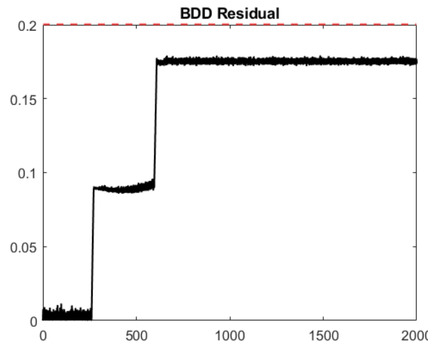

Fig. 8. The corresponding BDD residual under the generated attacks

Finally, we demonstrate the effectiveness of the proposed attack generation method through a time-series performance of generated feasible attacks in Fig. 7. The attacks caused the state estimator to output incorrect estimations of the pipeline flow and current node pressures. Then the controllers were driven to adjust the node pressures towards other malicious equilibrium points. Despite this, the stealthiness value, shown in Fig. 8, remained below the threshold, indicating that the attacks were able to remain undetected.

VI. Conclusions

This paper introduces a data-driven attack generative model for NPS vulnerability analysis. The framework consists of three interactive data-driven models: two discriminative models to evaluate attack signals and a generative model to generate feasible attacks. A new loss function is proposed to ensure successful attack generation.

Our future work involves providing the theoretical proof of the successful attack generation with the new loss function in (15). The boundary pursuit phenomenon observed in Fig. 5 suggests a potential for interactive training between the proposed attack generator and a supervised learning-based attack detector, which would lead to a complete automatic exploration of the vulnerability space and the development of a perfect attack detector for the system.

References

[1] A. Hobbs, "The colonial pipeline hack: Exposing vulnerabilities in us cybersecurity," in *SAGE Business Cases*. SAGE Publications: SAGE Business Cases Originals, 2021.

[2] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[3] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE transactions on automatic control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[4] G. Chen, Y. Zhang, S. Gu, and W. Hu, "Resilient state estimation and control of cyber-physical systems against false data injection attacks on both actuator and sensors," *IEEE Transactions on Control of Network Systems*, vol. 9, no. 1, pp. 500–510, 2021.

[5] T. Sui, Y. Mo, D. Marelli, X. Sun, and M. Fu, "The vulnerability of cyber-physical system under stealthy attacks," *IEEE Transactions on Automatic Control*, vol. 66, no. 2, pp. 637–650, 2020.

[6] X. Ren and Y. Mo, "Secure detection: Performance metric and sensor deployment strategy," *IEEE Transactions on Signal Processing*, vol. 66, no. 17, pp. 4450–4460, 2018.

[7] M. H. Shahriar, A. A. Khalil, M. A. Rahman, M. H. Manshaei, and D. Chen, "iattackgen: Generative synthesis of false data injection attacks in cyber-physical systems," in *2021 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2021, pp. 200–208.

[8] A. Khazraei, S. Hallyburton, Q. Gao, Y. Wang, and M. Pajic, "Learning-based vulnerability analysis of cyber-physical systems," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2022, pp. 259–269.

[9] A. N. Kolmogorov and A. T. Bharucha-Reid, *Foundations of the theory of probability: Second English Edition*. Courier Dover Publications, 2018.

[10] C. Godsil and G. F. Royle, *Algebraic graph theory*. Springer Science & Business Media, 2001, vol. 207.

[11] A. Osiadacz, *Simulation and analysis of gas networks*. Gulf Publishing Company, Houston, TX, 1987.

[12] H. Perez-Blanco and T. B. Henricks, "A gas turbine dynamic model for simulation and control," in *Turbo Expo: Power for Land, Sea, and Air*, vol. 78637. American Society of Mechanical Engineers, 1998, p. V002T03A002.

[13] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 1–33, 2011.

[14] Y. Zheng and O. M. Anubi, "Resilient observer design for cyber-physical systems with data-driven measurement pruning," in *Security and Resilience in Cyber-Physical Systems*. Switzerland: Springer, 2022, pp. 85–117.

[15] Y. Mo and B. Sinopoli, "False data injection attacks in control systems," in *Preprints of the 1st workshop on Secure Control Systems*, vol. 1, 2010.

[16] S. Zhao, H. Ren, A. Yuan, J. Song, N. Goodman, and S. Ermon, "Bias and generalization in deep generative models: An empirical study," *Advances in Neural Information Processing Systems*, vol. 31, 2018.