iAttackGen: Generative Synthesis of False Data Injection Attacks in Cyber-physical Systems

Md Hasan Shahriar*, Alvi Ataur Khalil*, Mohammad Ashiqur Rahman*,
Mohammad Hossein Manshaei*, and Dong Chen†
*Analytics for Cyber Defense (ACyD) Lab, Florida International University, FL, USA
†School of Computing and Information Sciences, Florida International University, FL, USA
*{mshah068, akhal042, marahman, mmanshae}@fiu.edu, †dochen@cs.fiu.edu

2021 IEEE Conference on Communications and Network Security (CNS) | 978-1-6654-4496-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CNS53000.2021.9705034

***Abstract*—As cyber-physical systems (CPSs) become an essential part of critical infrastructures and industries, their technological advancement creates a massive space for adversaries. Therefore, it is crucial to sufficiently explore the threat space to assess the systems' resiliency and plan for hardening. Moreover, due to any change in the cyber, physical, or operational level, CPSs often demand a re-analysis of potential threats. Threat analysis by conducting testbed experiments helps comprehend the attack potentiality but is infeasible to explore all attack space. Formal reasoning-based analytics are advantageous for threat analysis, especially for being noninvasive but provable. However, due to the convoluted features and non-linear nature of the system parameters, such formal models also become expensive in solving time, making them unscalable for larger systems. Hence, effective mechanism design is essential to augment the overall performance of the threat synthesis. This work proposes a threat analysis framework, named iAttackGen, where we train generative adversarial networks (GAN) models using the existing stealthy attack dataset (produced from testbed experiments or formal analysis) and generate more attack scenarios. We consider the smart power grid as the reference CPS and stealthy attacks as the threat model. Our evaluation results on standard IEEE bus systems prove iAttackGen's high accuracy and success rate in synthesizing potential threat vectors.**

***Index Terms*—False data injection attacks, generative adversarial networks, cyber-physical systems**

I. Introduction

Cyber-physical systems (CPSs) refer to the systems that effectively use modern sensors, computing mechanisms, and network technologies to integrate cyber and physical components effectively [1]. Although the unification of the Internet of things (IoT) with CPSs makes it smarter than ever, they also create huge space for the adversaries. Utilizing the targeted system's knowledge and states, adversaries can launch deceptive and influential stealthy attacks. False data injection (FDI) attacks is one of the prominent cyberattacks in CPSs, where adversaries often require the physical system's knowledge to construct attacks that can bypass existing bad/noisy data detection processes, keeping the altered measurements topologically harmonious with the non-attacked data collected by the supervisory control and data acquisition (SCADA) system to ensure stealthiness [2]–[4].

Many recent attacks against CPSs exhibit the need for security analysis and appropriate defense. Stuxnet, a computer worm, attacked over fifteen Iranian facilities, destroying over a thousand uranium enriching centrifuges [5]. BlackEnergy Malware successfully compromised the information systems of three energy distribution companies in Ukraine. They tripped 30 substations, leaving approximately 225,000 people in darkness for 1 to 6 hours [6]. Most recent, in 2021, a ransomware cyberattack on Colonial Pipeline, an American oil pipeline system, had all of its operations halted and paid $4.4 million within a few hours after the attack for the recovery [7].

Therefore, the comprehensive exploration of the threat space to assess a CPS's resiliency while dealing with any change in the cyber, physical, or operational level is crucial [8]. One group of threat analysis researchers conduct hands-on testing or testbed-based experiments to comprehend the attack potentiality (e.g., [9]). However, this approach is infeasible to explore the all attack space. Formal reasoning-based analytics have been proven to be advantageous for threat analysis due to their power of modeling the whole system and being provable and non-invasive (e.g., [10]). However, in the case of large systems, this approach also cannot scale due to the convoluted features and non-linear nature of the system parameters.

In Data-driven approaches, strategic decisions are taken based on data analysis and interpretation, where relationships between the system state variables are found without explicit system knowledge. These approaches can be trained to generate potential attacks much faster and their accuracy can be improved using rich historical data. One of the prominent data-driven approaches is generative adversarial networks (GAN), which has shown incredible generative capabilities by producing fake samples following almost the exact distribution of the training data [11]. Thus, we propose a framework, named *iAttackGen*, where we consider a customized GAN model and train it with an existing stealthy attack dataset to regenerate more attacks. As the power grids are an ideal example of CPSs and stealthy FDI attacks are well-researched in this domain, we consider them as the reference cases to evaluate the proposed framework. Our contribution in the paper can be summarized in the following points:

- We create an attack dataset for standard IEEE bus systems by leveraging a formal model of stealthy FDI attacks. We particularly consider the attacks on the state estimation (SE) process, the core module for the power grids.

- We design and implement the iAttackGen framework using GAN to generate both random and targeted stealthy threats. In the latter case, the framework considers the attacker's capability and target, i.e., a list of sensors or substations, as the conditions to the GAN.
- To generate optimized and fine-tuned threat vectors, we design a module, named harmonizer, to remove unnecessary noises from the GAN-generated samples.
- We evaluate the threat analysis success of iAttackGen on standard IEEE bus systems. The data and the codes related to the evaluation are also publicly available at https://sites.google.com/view/iattackgen.

Organization: We discuss the preliminary information in Section II. We introduce the proposed framework in Section III. In section IV, we discuss the technical details of the frameworks. In Section V, we explain the evaluation setup and dataset. The empirical analysis and findings are formulated in Section VI. The related works are discussed in Section VII. At last, we conclude the paper in Section VIII.

II. Preliminaries

In this section, we discuss the necessary preliminary information that helps to explain the iAttackGen framework.

A. State Estimation and Bad Data Detection

SE is one of the core parts of the SCADA system of CPSs, and bad data detection (BDD) is one of the main functions of SE. SE refers to a mathematical algorithm that processes raw measurement data collected from the remote sensors and estimates the system states [12]. SE and BDD help find the best states by considering only the compliant measurement data and eliminating outliers. In CPSs, if the measurement vector is z and the states vector is x, then their relationship can be presented as $z = h(x) + e$, where $h(x)$ represent the measurement functions that relate the measurement set to the states vector, and e is the set of measurement errors, which are assumed to be uncorrelated and follow normal distribution [12]. Linear estimation is done with the following equation: $\hat{x} = (H^T WH)^{-1} H^T Wz$, where $\hat{x}$, H, and W represent the most probable system states vector, the Jacobian matrix, and a diagonal weighting matrix, respectively. After system sates are estimated, residuals between measurements and estimated states are used to identify bad data. So, the estimation of measurements $\hat{z}$ is equal to $h(\hat{x})$ and the measurement residuals set r is calculated by $|z - \hat{z}|$. If we consider that threshold of residual as τ, any measurement z_i is regarded as bad data and removed from the SE procedure if $r_i > \tau$. , DC power flow is used in contingency analysis due to its simplicity, robustness, and high computation speed [13]. In DC approximation, bus voltage magnitudes are considered to be unity and state vector only contains the phase angles, making the h(x) a linear function. For the rest of the paper, we use $SE - BDD$ to indicate the SE and BDD together.

B. False Data Injection Attacks

In FDI attacks, sensor readings are compromised in such a way that the malicious data comply with the system topology and remain undetected by the BDD. Usually, random system errors of the measuring devices are responsible for the bad data. However, in the case of FDI attacks, malicious attackers deliberately construct and inject false data. Traditional BDD methods work efficiently for detecting bad data in SE; however, it becomes inefficient for detecting false data [14]. In FDI attacks, the goal of the attacker is to change the state variable $\hat{x}$ to $\hat{x_f}$ by modifying it with a malicious amount of c, i.e., $\hat{x_f} = \hat{x} + c$. To remain stealthy, false data a is maliciously injected into the measurement set z and the observed measuring is z_f, i.e., $\hat{z_f} = \hat{z} + a$. Now if the condition $a = h(c)$ holds, this false data a will be hidden from the traditional BDD, because the residual: $r = ||z_f - h(\hat{x_f})|| = ||(z + a) - h(\hat{x} + c)|| = ||z - h(\hat{x})||$. Thus, the data injection disappears from the residual, and the attack remains stealthy.

C. Generative Adversarial Networks

Goodfellow et al. introduced GAN, a novel way of training a generative model, where two "adversarial" neural networks compete against each other by playing a min-max game [15], replicating the real-world complex content like human-language, images, music, etc. A generative model G builds a mapping function from a noise distribution (p_z) to the distribution of the training data (p_{data}), which is presented as $G(z)$. On the other hand, the discriminator D outputs $D(x)$, the probability that a particular sample is from the real data set. Both G and D are trained simultaneously and they learn together. The Wasserstein GAN (WGAN) is an improvement to Goodfellow's GAN, where the objective function of the discriminator is presented by the earthmover (EM) distance between real and synthesized distributions [16]. The objective function of WGAN is defined as:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} [D(x)] - \mathbb{E}_{\boldsymbol{z} \sim p_z} [D(G(z))]$$

WGAN can be extended to a conditional WGAN (cWGAN) model if both the generator and discriminator are conditioned on some extra information y, which could be any kind of auxiliary data, such as class labels or data from other modalities. In this research, we also consider the attack attributes as the threat analysis condition. Thus, cWGAN learns to generate attack vectors considering the attacker's goal, accessibility, resources, and knowledge provided as the input. Therefore the application of cWGAN allows a controlled stealthy attack generation, where the objective function can be written as:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} [D(x|y)] - \mathbb{E}_{\boldsymbol{z} \sim p_z} [D(G(z|y))]$$

Here, x, y, z represent the targeted data, the condition, and the noise respectively. Thus, x and y are concatenated and considered as the input for the discriminator and z and y are the inputs for the generator. In this research, we particularly apply WGAN and cWGAN for random and targeted attack generation, respectively.

III. Proposed iAttackGen framework

As shown in Fig. 1, we logically separate our proposed iAttackGen framework into three modules: i) formal module, ii)

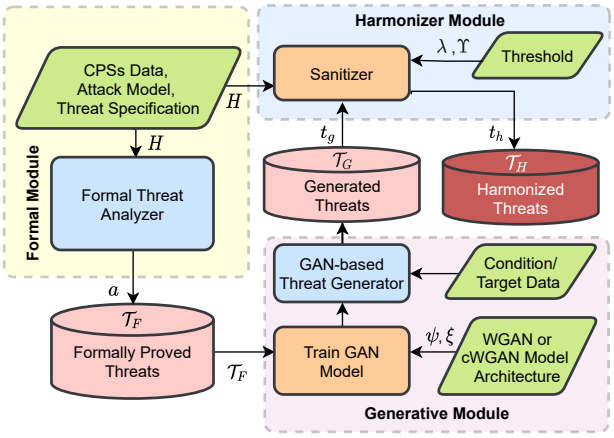


Fig. 1: Block diagram of proposed iAttackGen framework.

generative module, and iii) harmonizer module. The generative attack synthesis process starts with the formal module that takes the targeted CPSs data, attack model, and threat specifications as inputs and runs the formal threat analyzer to synthesize threat vectors. It designs the stealthy FDI threat generation as a constraint satisfaction problem (CSP). Successful executions of this module provide formally proven threat vectors, which are stored as $\mathcal{T}_F$, and later used by the generative module for the GAN's training.

The generative module is the core part of the iAttackGen framework, which performs two sequential steps. First, it trains the GAN model using the formally proven dataset $\mathcal{T}_F$. The GAN model can be either a regular WGAN to generate stealthy FDI threat vectors attacking any parts of the system, or it can be a cWGAN, which takes the attack attributes as the input conditions and generates threat vectors only attacking the attacker's targeted components. In this work, we consider the condition as a list of sensors or substations that the attacker has full access to and wants to generate the threats within them. As the core functions of WGAN and cWGAN are almost similar, we use GAN to represent both models in general. The GAN model architecture is designed considering the targeted system's topology. Once the GAN training reaches the equilibrium point, we utilize the trained generator G to synthesize new threat vectors and store them as generated threats $\mathcal{T}_G$. However, some threats may not strictly follow the FDI constraints due to the uncertainties in GAN's output. Hence, to ensure the stealthiness of the generated samples, we fine-tune them though the harmonizer module.

As the name implies, the harmonizer module enhances the stealthiness and effectiveness of the generated raw threat vectors through estimation and noise reduction. The core component is defined as a sanitizer, which takes input thresholds and regulates/estimates the threat vectors considering the system's topology and removes the minor/noisy components. The thresholds define the minimum range of injection value for any sensor in the harmonized threats. Finally, we store the harmonized threats as $\mathcal{T}_H$ and evaluate their performance. The following section provides a detailed overview of each module.

TABLE I: Modeling Parameters of Formal Constraints

Notation	Definition
b	Bus numbers in the grid.
l	Line numbers in the grid topology.
f_i	*from-bus*/starting bus of line i.
e_i	*to-bus*/ending bus of line i.
d_i	Admittance of line i.
P_i^L	Power flow of line i.
P_j^B	Power consumption of bus j.
θ_j	The state value, i.e. the phase angle at bus j.
$\mathbb{L}_{j,in}$	The sets of incoming lines to bus j.
$\mathbb{L}_{j,out}$	The sets of outgoing lines of bus j.
P_j^G	Generated power of bus j.
P_j^D	Load power of bus j.

IV. Technical Details

In this section, we elaborate the technical details of each module of the iAttackGen framework.

A. Formal Module

The formal module constructs the CSP considering the power system data, i.e., topology, states, etc. Attack attributes such as the attacker's goals, resources, targets, etc. are also considered in the CSP. In the power system, remote terminal units (RTU) and intelligent electronic devices (IED) belong to the local substations and report measurement data to the central energy management system (EMS) through the communication system. The sensor data contains the line power flow, bus consumption, switch status, etc. Usually, each transmission line has two sensors to measure the forward and backward line power flows, and each bus has one sensor to report the bus power consumption. Thus, any power system with b buses and l lines can have maximum $b + 2 * l$ sensors. EMS processes the system's topology matrix by utilizing reported sensor data and runs $SE-BDD$ algorithm estimate the states. A sophisticated attacker can have access to the sensor data and generate the topology matrix of the system. Thus, utilizing the resources, the attacker generates FDI threat vectors, where the goal is to deviate the SE by injecting the malicious data into the original sensor measurement.

The formalization of stealthy SE attacks is mainly adopted from existing works [17]. Table I presents the notations used for this modeling. Table II shows the formal constraints of the DC approximated power system's physical model, along with the constraints for FDI threat vectors. Equation (1) shows the relationship between the line power flow, line admittance, and bus phase angels during the system's pre-attack scenario. Bus power consumption is the difference between bus generation and bus load, as shown in (2) and (3). The injected false data in the power flow measurement of line i, ΔP_i^L, is calculated using (4), where $\Delta\theta_j$ is the injection in the sates of bus j. The constraint for the injection ΔP_j^B in the power consumption of bus j is shown in (6). The final malicious measurements of compromised line power flow and bus consumption are expressed as (5) and (7). A critical feature of the FDI threat vector for a power system is that if one transmission line's

TABLE II: Formalization of False Data Injection Attack

Line Power Flow and Bus Power Consumption:

$$\forall_{1\le i\le l}\quad P_i^L = d_i(\theta_{f_i} - \theta_{e_i}) \tag{1}$$

$$\forall_{1\le j\le b}\ P_j^B = \sum_{i\in\mathbb{L}_{j,in}} P_i^L - \sum_{i\in\mathbb{L}_{j,out}} P_i^L \tag{2}$$

$$\forall_{1\le j\le b}\quad P_j^B = P_j^D\ - P_j^G \tag{3}$$

Attacked Line Power Flow, Bus and Load Data:

$$\forall_{1\le i\le l}\quad \Delta P_i^L = d_i(\Delta\theta_{f_i} - \Delta\theta_{e_i}) \tag{4}$$

$$\forall_{1\le i\le l}\quad \bar{P}_i^L = P_i^L + \Delta P_i^L \tag{5}$$

$$\forall_{1\le j\le b}\ \Delta P_j^B = \sum_{i\in\mathbb{L}_{j,in}} \Delta P_i^L - \sum_{i\in\mathbb{L}_{j,out}} \Delta P_i^L \tag{6}$$

$$\forall_{1\le j\le b}\ \bar{P}_j^B = P_j^B + \Delta P_j^B \tag{7}$$

forward line power flow is compromised , then the backward line power flow also must be compromised with the same magnitude to make the attack stealthy.

Each threat vector contains the information on which sensors need to be compromised and by what amount. Thus, all the values of ΔP_i^L and ΔP_j^B together represent a complete threat vector a. The zero injection value for any sensor indicates that the sensor does not need to be compromised to launch the attack. Only the sensors with non-zero injection values are to be compromised. All the synthesized threats are formally proved and added to $\mathcal{T}_F$. As the system has m sensors, a threat vector is also an m dimensional vector. Let's assume that the formal module generates k threats vectors; thus, the dataset $\mathcal{T}_F$ has k rows and m columns.

B. Generative Module

The generative module does two sequential tasks: training the GAN model and synthesizing the threat vectors. Depending on the types of threat generation, WGAN or cWGAN is chosen as the GAN model.

1) Random Threat Generation: In this subsection, we present the architecture and training of the WGAN model and the process of random threat generation.

GAN Architecture: In WGAN, the input layer of the generator G takes noise as input. The dimension of the noise (latent space) depends on the system topology, which we define as p. As the measurement vector is topologically dependant on the state vector ($\in \mathcal{R}^b$), we consider the noise dimension p equal to b. The number of nodes in the output layer of G is the same as the dimension of the threat vector, which is m. Similarly, the input layer of the discriminator (D) has m nodes. The output layer of D has only one node, which outputs the probability of the provided sample threat vector of being stealthy. The number of nodes and hidden layers of G and D depend on the size of the targeted system, i.e., the IEEE bus system. We represent these model parameters for G and D, which also include the learning rate, optimizer, etc. by ψ and ξ, respectively.

Algorithm 1: Training algorithm of WAGN model.

initialize ξ = discriminator's parameters;
initialize ψ = generator's parameters;
while ψ *has not converged* **do**
  for n_{discri} *steps* **do**
    Sample real examples $x^{(1)}, x^{(2)}, .., x^{(m)}$ from $\mathcal{T}_F$
    Sample noises $z^{(1)}, z^{(2)}, .., z^{(m)}$ from $p_g(z)$
    $g_\xi \leftarrow \nabla_\xi \left[\frac{1}{m}\sum_{i=1}^m f_\xi\left(x^{(i)}\right) - \frac{1}{m}\sum_{i=1}^m f_\xi\left(g_\psi\left(z^{(i)}\right)\right)\right]$
    Update: $\xi \leftarrow \xi + \alpha \cdot \mathrm{RMSProp}\left(\xi, g_\xi\right)$
    Clip: $\xi \leftarrow \mathrm{clip}(\xi, -c, c)$
  end
  Sample noises $z^{(1)}, z^{(2)}, .., z^{(m)}$ from noise distribution $p_g(z)$
  Gradient: $g_\psi \leftarrow -\nabla_\psi \frac{1}{m}\sum_{i=1}^m f_\xi\left(g_\psi\left(z^{(i)}\right)\right)$
  Update: $\psi \leftarrow \psi - \alpha \cdot \mathrm{RMSProp}\left(\psi, g_\psi\right)$
  for *from* $i = 1$ *to* $n_{harmonize}$ **do**
    Generate a random noise vector, z
    Generate a raw threat sample, $t_g \leftarrow g_\psi(z)$
    Harmonized threat, $t_h \leftarrow H(H^TH)^{-1}H^Tt_r$
    Gradient: $g_\psi \leftarrow \nabla_\psi MeanSquareError(t_r, t_h)$
    Update: $\psi \leftarrow \psi - \eta \cdot \mathrm{RMSProp}\left(\psi, g_\psi\right)$
  end
end

Training and Generation: We train WGAN model on $\mathcal{T}_F$ following the approach as shown in Algorithm 1 with the weight clipping technique. The training is divided into three parts. First, D is trained using a batch of real threat samples and then WGAN-generated fake samples. Secondly, G is trained on the synthetic threats and the weights are updated with respect to D's prediction. As WGAN learns the data distribution, the training algorithm's last loop is not something theoretically mandatory. However, we add this extra loop to accelerate the learning of stealthy nature of the threat vectors by G. This part generates few threat vectors, estimates their stealthy shape, and retrain G with the mean square error between the generated threats and the estimated threats. Once the training is over, for generating l threat instances, we feed an $l \times p$ dimensional Gaussian noise to G and it outputs $l \times m$ dimensional threat vector data. We stored the generated threat dataset as $\mathcal{T}_G$.

2) Targeted Attack Generation: We use cWGAN to generate targeted threats. Depending on the attacker's goal, we consider two cases of the cWGAN. In the first case, the goal is to create threats compromising sensors within the given set of sensors as a condition. In the second case, the condition is a set of substations/buses, and the compromised sensors only belong to them. In these two cases, the generative models are expressed as cWGAN-Sen and cWGAN-Sub as the conditional data are the set of sensors, and substations, respectively.

GAN Architecture: In the conditional models, the input of both G and D take the conditional data along with the regular input. The condition of cWGAN-Sen is an m-dimensional encoded binary data, where 1 means the sensor can be considered in the threat generation, and 0 means not. The condition

TABLE III: Architecture of GAN models

Net	Layer	Random Attack	Targeted Attack	
		WGAN	*cWGAN-Sen*	*cWGAN-Sub*
G	*Input*	$\mathcal{R}^p$	$\mathcal{R}^{m+p}$	$\mathcal{R}^{b+p}$
	Output	$\mathcal{R}^m$	$\mathcal{R}^m$	$\mathcal{R}^m$
D	*Input*	$\mathcal{R}^m$	$\mathcal{R}^{m+m}$	$\mathcal{R}^{b+m}$
	Output	$\mathcal{R}^1$	$\mathcal{R}^1$	$\mathcal{R}^1$

of cWGAN-Sub is also a b-dimensional binary data, where each binary element indicates the corresponding substation. Similarly, 1 or 0 in the conditional data represents the attacker's access to that substation, and thus, whether sensors from that substations should be considered or not in the threat generation. For both cases, the dimension of the input noise is the same as the number of buses. These input and output data dimensions are summarized in Table III.

Training and Generation: We train cWGAN models following a similar algorithm as WGAN. Additionally, here we concatenate the condition with the noise as the input of G, and with the samples in case of D. In this case, we need to generate the conditional data for training from the formally proved threat dataset $\mathcal{T}_F$. In the case of cWGAN-Sen, the conditional data is the binary encoding of the threat vectors. The measurements having non-zero and zero magnitudes are encoded as ones, and zeros, respectively. Similarly, in generating the conditional data of cWGAN-Sub, we assign one for any substation if at least one sensor from that substations is considered in the threat vector, otherwise zero. After successful training, we provide similar conditional data to each model and analyze the generated threat vectors' performance.

C. Harmonizer Module

To ensure the stealthiness of the GAN generated threat vector dataset $\mathcal{T}_G$, we apply the necessary calibration on the samples. The harmonizer fine-tunes the generated raw threat vectors. As shown in Algorithm 2, the harmonizer module primarily does two operations. Firstly, it estimates the generated threats using the system's topology matrix H and then amends (deletion/correction) the non-compliant data. As shown in Section II, a generated FDI threat vector, t_g can be expressed as Hc, where c is the malicious injection in state data. Hence, the best estimation of c can be defined as $\hat{c}$, where $\hat{c} = (H^T H)^{-1} H^T t_g$. Thus, the harmonized (estimated) threat vector $t_h = H\hat{c}$ as shown in the algorithm

Although the GAN model generates samples according to the real data distribution, some sensors can be assigned with insignificant injection amounts, which may not significantly influence the attack impact. However, compromising such sensors adds up the cost of the attack. Thus, the sanitizer also controls the significance of the attack points. Let's assume the mean and standard deviation of any threat vector are μ, and σ, respectively. A factor λ is used to calculate two threshold parameters η_{min}, and η_{max}, where $\eta_{max} = \mu + \lambda\sigma$, and $\eta_{min} = \mu - \lambda\sigma$. The thresholds define the range beyond which the injection amounts are considered significant; otherwise, they are considered insignificant and removed.

Algorithm 2: Sanitization Algorithm $(\lambda, \Upsilon, \mathcal{T}_G)$

```
initialize harmonized threats dataset, T_H;
for each generated threat vector t_g ∈ T_G do
    for Harmonization repeat, j = 1 to Υ do
        Harmonized sample, t_h ← H(H^T H)^{-1} H^T t_g
        μ = Mean(t_h), σ = StandardDeviation(t_h)
        Threshold, η_max = μ + λσ, and η_min = μ − λσ
        for each element t^(i) ∈ t_h do
            if η_min ≤ t^(i) ≤ η_max then
                Remove insignificant sensor, t^(i) ← 0
            end
        end
        if t_h is stealthy then
            Update harmonized dataset, T_H.insert(t_h)
            break
        end
        else
            Repeat with the same attack vector, t_g ← t_h
        end
    end
end
```

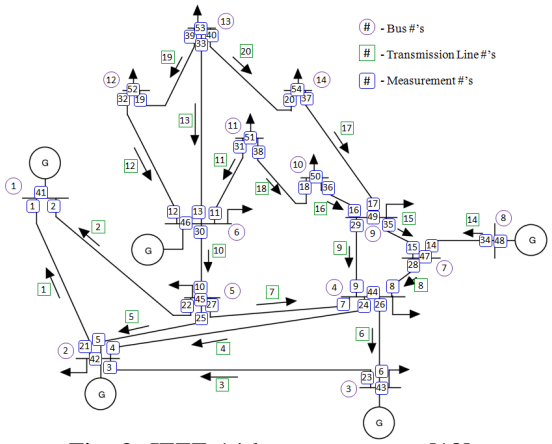


Fig. 2: IEEE 14-bus test system [18].

The cleansing of insignificant elements from the raw threats keeps the sensors only with the notable injection amount contributing to the attack's intent. The value of λ controls the number of cleaned sensors in the harmonized threats. This cleansing makes the attack less expensive with respect to the number of sensors to be compromised but sacrificing less on the attack impacts by cleansing the insignificant sensors. By adjusting the threshold value λ, the sanitizer can also control the number of sensors to be altered/accessed to launch an attack. However, such removal sometimes disrupts the harmonized threat vectors' compliance/stealthiness; thus, further harmonization might be needed to ensure the stealthiness after such removal. Therefore, we introduced another parameter of Υ, which is the maximum repetition of the harmonization process on a threat vector. After the systematic application of these operations, we get the harmonized version of the generated samples stored as $\mathcal{T}_H$. Now, the dimension of $\mathcal{T}_H$ depends on the successful harmonization of $\mathcal{T}_G$. If we assume that among l generated threats, only h samples remain stealthy after harmonization, then the dimension of $\mathcal{T}_H$ is $h \times m$, and the success rate of the harmonizer is $\frac{h}{l} * 100\%$.

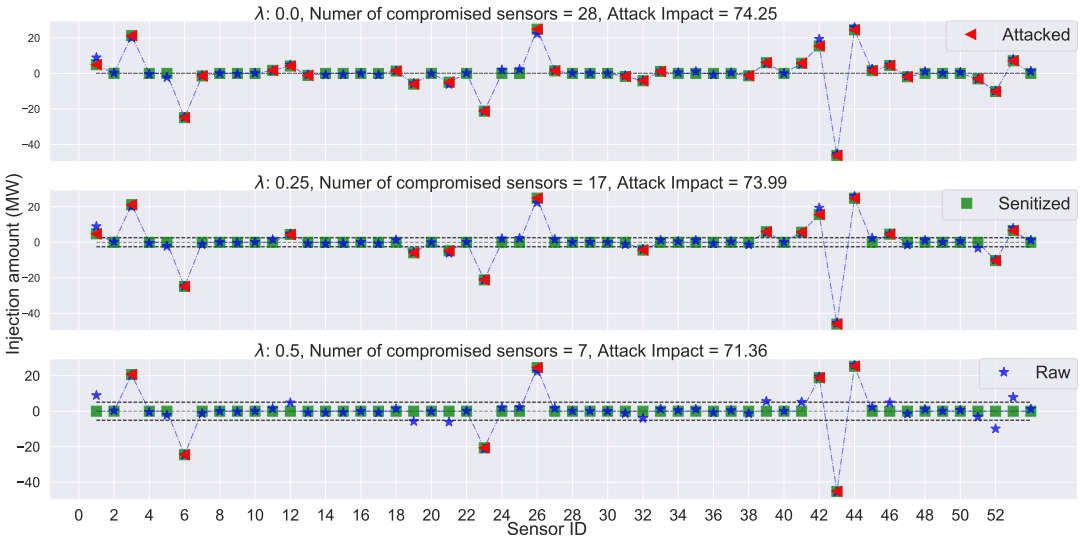


Fig. 3: Impact of threshold factor λ in threat sanitation process.

D. A 14-Bus Case Study

We conduct a case considering the standard IEEE 14 bus system, as shown in Fig. 2. It has 20 lines, 14 buses, and a maximum of 54 (i.e., 2*20+14) sensors. Fig. 3 shows how harmonization reshapes the raw threat vector by eliminating and amending the sensors for different λ. As the figure illustrates, the WGAN generated raw attack vector and three stages of sanitization for a λ of 0, 0.25, and 0.5, respectively. In the first case (λ = 0), nothing is removed, and 28 sensors with the non-zero injections are considered to be attacked. The impact of such a raw threat is 74.25. Later, for λ as 0.25, any estimated injections within the range $\mu \pm 0.25\sigma$ are replaced with zeros. Thus, the number of non-zero injections in the sanitized threat is reduced to 17 only. Even though the number of compromised sensors is almost halved, the AI is still 73.99. Similarly, for λ of 0.50, another ten non-zero sensors fall within the increased range and get removed from the threat. In this case, the number of compromised sensors is only 7, and the impact is still 71.36. Therefore, by increasing λ from 0.00 to 0.50, the number of compromised sensors is reduced by 75% but attack impact is reduced by only 3.89%.

V. Evaluation Setup

This section presents the experimental setup and necessary evaluation metrics to assess our proposed iAttackGen framework's performance. We implement the framework on the standard IEEE 30 bus system [19]. It has 41 lines, 30 buses, and a maximum of 112 (2*41+30) sensors.

A. Formally Proved Dataset $\mathcal{T}_F$

We run the formal method considering all the sensors are accessible to the attacker, and he/she may compromise a maximum of five substations at a time. After the successful executions of the formal model, we generate 120 FDI threat vectors. Thus, the dimension of the formal dataset $\mathcal{T}_F$ is 120x112. The dataset has 112 features, where the first 41 features represent the forward line power flows, the next 41 features are for the backward line power flows of the transmission lines, and the last 30 features hold the bus power consumption data. Thus, there is a negative correlation between each pair of the forward and backward line power flows, to come up with an optimal choice of parameters.

TABLE IV: GAN Architecture

GAN Architecture	**Generator** G	**Discriminator** D
Model	ANN	ANN
Layers	4	4
Nodes (WGAN) 14 Bus	14, 20, 40, 54	54, 25, 10, 1
Nodes (WGAN) 30 Bus	30, 56, 80, 112	112, 64, 32, 1
Nodes (CWGAN-Sen) 30 Bus	142, 130, 120, 112	224, 96, 68, 1
Nodes (CWGAN-Sub) 30 Bus	60, 64, 96, 112	142, 68, 32, 1
*Activation**	LR, LR, LR, LN	LR, LR, LR, SG
Library	PyTorch	
Optimizer	RMSprop	
Learning Rate	5×10^{-5}	
Max Epoch	10,000	
Dimension of Noise	14 (14 Bus), 30 (30 Bus)	

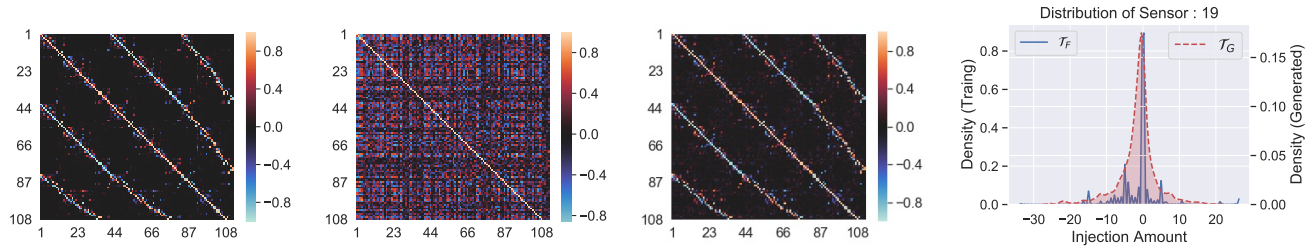
*LR : Leaky-Relu, LN : Linear, SG : Sigmoid

B. GAN Model Architecture

The architecture of the GAN models depends on the considered system's topology. All three GAN models have similar architectures and training set up, except the number of nodes in different layers of G and D. Table IV shows the GAN models' architectures and training parameters. Both the generator and discriminator are four layers of artificial neural networks (ANN). We implement the models in Python using the PyTorch library.

C. Harmonizer Parameters

To evaluate the necessity and the impacts of the harmonizer module, we use different threshold factor values λ and maximum repeat Υ. We analyze the harmonizer module's performance for λ of 0, 0.25, 0.5, 0.75, and 1.00. Similarly, to show how repetitive harmonization helps in fine-tuning the raw threats, we use the values of 0, 1, 2, 3, 4, and 5 as Υ, where 0 indicates no harmonization.



(a) (b) (c) (d)

Fig. 4: Correlation heatmap of the (a) WGAN training data, (b-c) WGAN-generated data at epoch 0 and 1000, respectively, and (d) distribution of the sensor-wise injection for sensor 19 in $\mathcal{T}_F$ and $\mathcal{T}_G$.

D. Evaluation metrics

In this subsection, we define the metrics used to evaluate the iAttackGen's performance.

Success Rate (SR): A threat vector is considered stealthy if none of the compromised sensors becomes outlier during the injection. Thus, we consider the harmonization of a raw threat vector successful if it's the harmonized version remains stealthy. Thus the SR can be defined as the following:

$$SR = \frac{\textit{\# of stealthy harmonized threats}}{\textit{\# of total generated threats}} \times 100$$

Attack Impact (AI) is defined by the magnitude of deviation in the measurement estimation due to the attack. It is the $l-2$ norm of threat vector t_h. $AI = |Z_{org} - Z_{att}| = |t_h|$, where Z_{org} and Z_{att} are the original and attack measurement vectors. AI indicates the severity of a threat.

Compliance refers to the fraction of compromised elements (sensors/substation) that are intended to be compromised. It helps to find if the model is considering sensors outside the intended boundary set by the condition.

$$Compliance = \frac{\textit{\# of compromised elements in condition}}{\textit{\# of total compromised elements}}$$

Utilization refers to the fraction of elements that are compromised among the total number of elements that are supposed to be compromised. It helps understand if assets are fully utilized or not.

$$Utilization = \frac{\textit{\# of compromised elements condition}}{\textit{\# of total elements in condition}}$$

VI. Evaluation Results and Discussion

In this section, we evaluate and analyze the iAttackGen's performance in three steps. Firstly, we analyze the performance of GAN models in training and generating new samples. Later, we analyze different harmonization parameters' impact. Lastly, we show the performance of cWGAN with respect to compliance and utilization.

A. Evaluation of Random Attack Generation

1) Training of WGAN: In this part, we specifically focus on the WGAN model and it's learned correlation matrix of the different features. Fig. 4(a) shows the actual correlation matrix of the training data ($\mathcal{T}_F$). Fig. 4 (b-c) show the correlation matrices of WGAN generated fake data at different phases of the training process. Fig. 4(b) shows random noise at epoch 0 as WGAN does not know the distribution of the features and their dependencies. Fig. 4(c) show the correlation at epoch 1000, which looks almost the same as Fig. 4(a), indicating the model's accuracy in generating realistic fake samples.

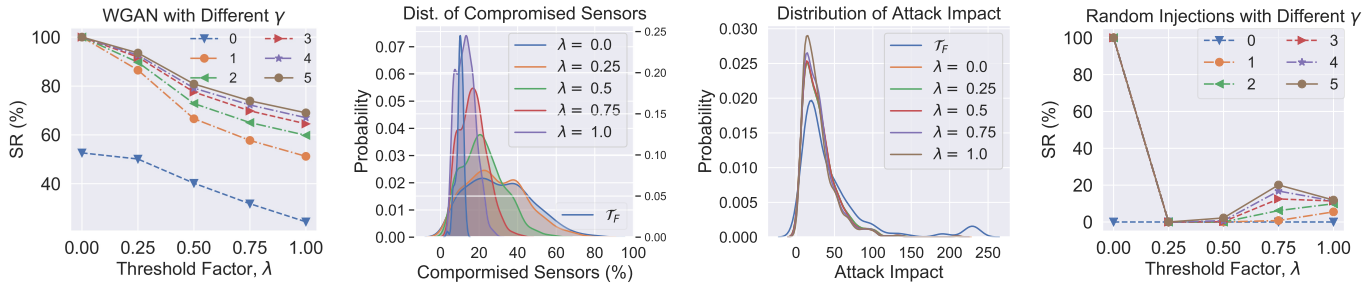
2) Distribution of $\mathcal{T}_G$: In this part, we evaluate the training of WGAN for learning the generalized distribution of features. After training the WGAN model, we synthesize 1000 sample threats using the trained generator. Among the 112 features, we randomly select one sensor: 19, and plot it's distributions for both $\mathcal{T}_F$ and $\mathcal{T}_G$. As Fig. 4(d) illustrates, the distributions of the sensors in $\mathcal{T}_F$ are incomplete as it contains a limited number of threat vectors and spikes at different points indicate specific injection amounts of the sensors. However, the distributions of the generated dataset $\mathcal{T}_G$ look almost Gaussian and uniformly cover a wide range of injections. Thus, WGAN is can generalize the distribution although it is trained with incomplete training data.

B. Evaluation of Harmonizer

The harmonizer reshapes the raw attack vectors based on λ and Υ. Fig. 5 shows the impact of these parameters on different aspects of the harmonized threat vectors.

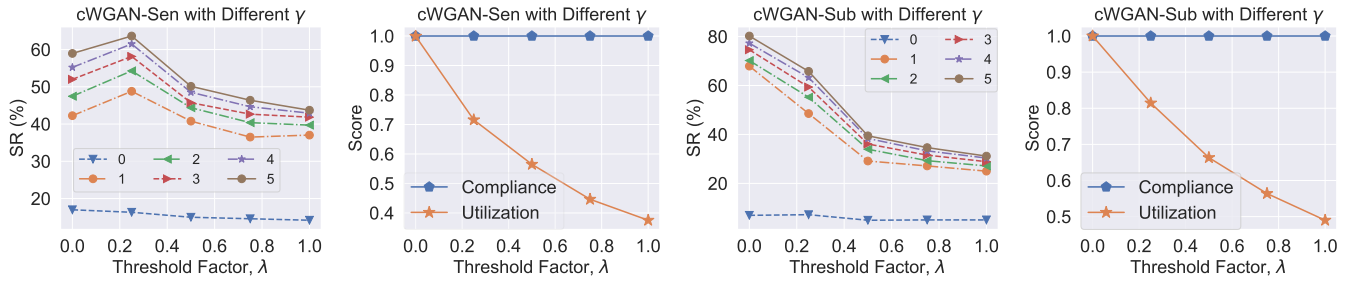
Impact on Success Rate: Fig. 5(a) shows the impact of λ and Υ on the harmonizer's SR. The figure illustrates, as λ increases, the harmonizer removes more elements with the injection amounts within (η_{min}, η_{max}). Thus, a few of the threats lose harmony, and the SR decreases. However, repetitive harmonization improves the SR notably. As shown in the figure, with no harmonization (Υ is 0), the SR is too low. Thus, WGAN generated raw threats that are not perfectly clean, which mandates their sanitization. As we introduce harmonization, the SR quickly increases to 100% with λ equals 0.00. Similarly, in case of λ being equal to 1.00, as we increase the value of Υ from 1 to 5, the SR increases by 25% up to 50%. Due to the multiple harmonizations, the unpolished trimmed threats converge towards the stealthy versions.

Impact on Compromised Sensors: The cost of a threat depends highly on the number of compromised sensors. This cost can be controlled by varying the λ, and removing a few sensors by replacing their injections with zeros. The higher λ is, the less is the number of total compromised sensors. Fig. 5(b) shows the distribution of the percentage of compromised sensors in the harmonized threats for different values



(a) (b) (c) (d)

Fig. 5: Impacts of harmonizer (λ and γ) on WGAN generated threats w.r.t., a) success rate, b) percentage of compromised sensors, and c) attack impact, and on threats with random noise injections w.r.t., d) success rate.



(a) (b) (c) (d)

Fig. 6: Performance on targeted attack generation of (a-b) cWGAN-Sen, and (c-d) cWGAN-Sub.

of λ. The figure illustrates as the training data ($\mathcal{T}_F$) contains the threats, wherein average 20-30% sensors are considered. However, the distribution of the harmonized threats without any cleaning (λ as 0) spans a vast range, which is approximately 20% to 100%, which are highly expensive for the sensor accessibility. Moreover, it might not be feasible for an attacker to compromise almost 100% of the sensors. However, such a percentage can be reduced by increasing the sanitization threshold range. As the figure shows, the distribution gets shifted towards zero as λ increases. Even in the case of λ as 1.00, on average, 15% of the sensors are considered in the harmonized threats. Thus, by tuning λ, we can give the threats a perfect shape according to the attacker's goals and resources.

Impact on Attack Impact: Fig. 5(c) shows the average AI of the harmonized threat vectors for different values of λ. The figure illustrates, even though more sensors are removed from the harmonized threats with increasing λ, there is very little change in the distribution of AI. As shown in Fig. 5(b-c), initially (λ as 0), the average AI of the threats is 50, and the average compromised sensors are approximately 50%. However, for λ as 1.00, though the average compromised sensors are only 20%, the average AI remains the same. Hence, this evaluation proves that the harmonization module successfully removes the insignificant sensors from the raw threat vectors without hampering the intensity of them.

Harmonization of Random Injection Vectors: Lastly, we evaluate the performance of the harmonizer in the absence of WGAN. We generate threats using a random number generator. Being just random numbers, as Fig. 5(d) illustrates, none of the threats are stealthy without any harmonization ($\Upsilon = 0$). Even though one-time harmonization can make them stealthy, if we increase λ to make the threats feasible and less expensive, SR becomes almost 0%. This analysis mandates the application of generative models in threat synthesis.

C. Evaluation of Targeted Attack Generation

Performance of cWGAN-Sen: Similar to the evaluation of the WGAN model, we evaluate the SR of cWGAN-Sen. Fig. 6(a) shows that the SR of cWGAN-Sen also follows a similar trend as WGAN. By implementing repetitive harmonization, the SR can be increased from 7% to approximately 55%. Besides, as we explicitly add the condition in the harmonization process, as Fig. 6(b) illustrates, the modified threats show a 100% compliance in considering the sensors in the threat vectors. Thus, none of the sensors are considered outside the provided list of sensors as a condition. Similarly, the utilization of the initial threats is also 100%. However, as λ increases, more sensors are removed from the threats; thus, the utilization decreases proportionally.

Performance of cWGAN-Sub: In the evaluation of cWGAN-Sub, we provide the list of substations as the condition. As each substation has multiple sensors, the cWGAN-Sub model gets more flexibility in selecting sensors from different substations. Thus, Fig. 6(c) shows a little better SR in the case of cWGAN-Sub than cWGAN-Sen. In this case, the SR can go as high as 75%. Similar to cWGAN-Sen, the compliance, and utilization graphs show a similar pattern for cWGAN-Sub. As shown in Fig. 6(d), the compromised sensors in the harmonized threats only come from the permitted substations, making the

compliance 100%. The utilization also starts from 100% and decreases as λ increases.

VII. Related Work

Even though GAN is primarily developed focusing on computer vision, researchers are investigating the application of GAN in other domains, including the CPSs. GAN is used to improve the accuracy of detecting malicious samples in the IDS. Ferdowsi et al. used GAN to design a decentralized IDS for the IoT realm to detect anomalies [20]. In [21], Chhetri et al. proposed a conditional GAN (cGAN)-based model to analyze the security requirements between the cyber and physical domains in a CPS.

A few of the researches focus on the generation of malicious samples. Lin et al. proposed IDSGAN generating the adversarial attacks, which can bypass the black box IDS [22]. Hu et al. proposed a similar framework named MalGAN to generate malware samples against black-box ML-based IDS [23]. Shahriar et al. proposed a GAN-based approach to create a synthetic network attack dataset from the existing data [24]. Ahmadian et al. presented GAN based false load data generator in [25], which only paper focuses on the false sensor data generation. However, they did not consider the sensors' physical relationships. In regards to resisting cyber attacks leveraging GAN, Li et al. proposed a GAN based cyber-physical model to defend against FDI attacks [26].

Most of the existing works investigated the application of GAN for CPSs network datasets. Even though some of them considered sensor datasets, the application of GAN for the FDI threat generation is less explored. Mohammadpourfard et al. proposed the generation of FDI attacks using cGAN in [27]; however, they didn't consider attacker's capability. The conditions only specified the states to be manipulated, not the measurements that need to be altered. They also didn't consider attack data for training. The malicious data for network packets, i.e., IP address, protocol type, flags, etc., can only be generated considering the statistical distribution. However, in the case of stealthy attack synthesis, the stealthiness must be ensured based on the CPSs topology. Our study shows that mere GAN cannot generate perfect stealthy samples; instead, these samples need to be further refined.

VIII. Conclusion

Analyzing the threat space is the essential technique to assess the vulnerability and increase the robustness of CPSs-based critical infrastructures. Constrains satisfaction-based methods are highly computationally expensive and lose scalability in the case of dynamic complex systems. However, data-driven approaches are showing promising results in solving complex problems in realtime. In this paper, we propose a GAN-based intelligent attack generation framework, titled iAttackGen. The framework is trained real or formally synthesized threats dataset and synthesizes new raw threats. A harmonizer module ensures the stealthiness of the samples by adjusting them according to the system's topology. Our evaluation results on the standard IEEE bus system show a high success rate in random and targeted threat generation.

References

[1] S. Zeadally and N. Jabeur, *Cyber-physical system design with sensor networking technologies*. Inst. of Engineering and Technology, 2016.

[2] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *ACM CCS*, 2009.

[3] M. Rahman, M. Shahriar, M. Jafari, and R. Masum, "Novel attacks against contingency analysis in power grids," *arXiv preprint arXiv:1911.00928*, 2019.

[4] M. Jafari, M. H. Shahriar, M. A. Rahman, and S. Paudyal, "False relay operation attacks in power systems with high renewables," *arXiv preprint arXiv:2102.12041*.

[5] D. Albright, P. Brannan, and C. Walrond, "Stuxnet malware and natanz: Update of isis december 22, 2010 report," *ISIS*, 2011.

[6] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, "Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid," in *4th ICS-CSR 2016*.

[7] C. B. Kelly, "Cyber attack shuts down u.s. fuel pipeline 'jugular,' biden briefed," 2021, accessed: 2021-06-17.

[8] N. I. Haque, M. H. Shahriar, M. G. Dastgir, A. Debnath, I. Parvez, A. Sarwat, and M. A. Rahman, "A survey of machine learning-based cyber-physical attack generation, detection, and mitigation in smart-grid," in *52nd North American Power Symposium (NAPS)*. IEEE, 2021.

[9] A. Barua, "Hall spoofing: A non-invasive dos attack on grid-tied solar inverter," in *29th USENIX Security Symposium*, 2020.

[10] M. A. Rahman, E. Al-Shaer, and R. Kavasseri, "A formal model for verifying the impact of stealthy attacks on optimal power flow in power grids," in *ACM/IEEE ICCPS*, Apr. 2014.

[11] C. Li, S. Wang, P. S. Yu, L. Zheng, X. Zhang, Z. Li, and Y. Liang, "Distribution distance minimization for unsupervised user identity linkage," in *27th ACM CIKM*, 2018.

[12] A. Abur and A. G. Exposito, *Power system state estimation: theory and implementation*. CRC press, 2004.

[13] D. Van Hertem, J. Verboomen, K. Purchala, R. Belmans, and W. L. Kling, "Usefulness of dc power flow for active power flow analysis with flow controlling devices," in *8th IEE ACDC*, 2006.

[14] R. Xu, R. Wang, Z. Guan, L. Wu, J. Wu, and X. Du, "Achieving efficient detection against false data injection attacks in smart grid," *IEEE Access*.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*.

[16] Q. Wang, X. Zhou, C. Wang, Z. Liu, J. Huang, Y. Zhou, C. Li, H. Zhuang, and J. Cheng, "Wgan-based synthetic minority over-sampling technique: Improving semantic fine-grained classification for lung nodules in ct images," *IEEE Access*.

[17] M. Rahman, E. Al-Shaer, and R. Kavasseri, "A formal model for verifying the impact of stealthy attacks on optimal power flow in power grids," in *2014 ACM/IEEE ICCPS*. IEEE, 2014.

[18] M. A. Rahman, E. Al-Shaer, and R. Kavasseri, "Impact analysis of topology poisoning attacks on economic operation of the smart power grid," in *ICDCS*, Jul. 2014.

[19] "Ieee 30-bus system," *www.icseg.iti.illinois.edu/power-cases/*.

[20] A. Ferdowsi and W. Saad, "Generative adversarial networks for distributed intrusion detection in the internet of things," 2019.

[21] S. R. Chhetri, A. B. Lopez, J. Wan, and M. A. Al Faruque, "Gan-sec: Generative adversarial network modeling for the security analysis of cyber-physical production systems," in *2019 DATE*, 2019.

[22] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," *arXiv*, 2018.

[23] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," *arXiv preprint arXiv:1702.05983*, 2017.

[24] M. H. Shahriar, N. I. Haque, M. A. Rahman, and M. Alonso, "G-ids: Generative adversarial networks assisted intrusion detection system," in *IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020.

[25] S. Ahmadian, H. Malki, and Z. Han, "Cyber attacks on smart energy grids using generative adverserial networks," in *IEEE GlobalSIP*, 2018.

[26] Y. Li, Y. Wang, and S. Hu, "Online generative adversary network based measurement recovery in false data injection attacks: A cyber-physical approach," *IEEE Transactions on Industrial Informatics*, 2019.

[27] M. Mohammadpourfard, F. Ghanaatpishe, M. Mohammadi, S. L., and M. Pechenizkiy, "Generation of false data injection attacks using conditional generative adversarial networks," in *2020 IEEE PES ISGT-Europe*.