Grenze International Journal of Engineering and Technology, June Issue

Enhancing Industrial Protocol Security with GAN-based Fuzz Testing and Self-Attention Mechanism

Elvis Mondal
CIeNET Technologies, Warren, Michigan
Email: elvismondal@hotmail.com

***Abstract*— This study introduces a novel deep learning framework for identifying vulnerabilities in Industrial Control Protocols (ICPs) through intelligent fuzz testing. Leveraging a Wasserstein GAN with Gradient Penalty (WGAN-GP) and self-attention mechanisms, the proposed model generates realistic yet diverse protocol message sequences to effectively detect potential security threats. By bypassing the need for manual protocol specifications, the approach improves test case generation and accelerates anomaly detection. Extensive evaluations on MQTT and Modbus- TCP protocols demonstrate superior performance in both accuracy and vulnerability detection compared to traditional and deep learning-based fuzzing methods.**

***Keywords*— Industrial control protocols · Fuzz testing · Generative adversarial networks · Self-attention mechanism · Cybersecurity.**

I. INTRODUCTION

Industrial Control Systems (ICS) form the backbone of critical infrastructure across various sectors including energy, manufacturing, and transportation. With the advent of Industry 4.0 [1] and the increasing integration of operational technology with information technology networks, these systems have become increasingly vulnerable to cyber threats. Industrial Control Protocols (ICPs) serve as the communication foundation within ICS environments, making them prime targets for potential attackers seeking to disrupt industrial operations.

Traditional security assessment methods for ICPs often rely on manual analysis of protocol specifications and the development of custom test cases. This approach presents several significant limitations, including high expertise requirements, lengthy testing cycles, and lack of universality across different protocol types. Fuzz testing has emerged as a prominent technique for vulnerability discovery in ICPs, but conventional fuzzing methods struggle with the complexity and diversity of industrial communication patterns.

The application of machine learning techniques to fuzz testing represents a promising direction for addressing these challenges. However, existing deep learning approaches for fuzzing often suffer from training instability, limited diversity in generated test cases, and inefficient resource utilization. These limitations be- come particularly pronounced in industrial environments where protocols must maintain real-time performance while ensuring operational safety.

This paper presents HexGANFuzzer, a novel fuzzing framework that com- bines Wasserstein Generative Adversarial Networks with self-attention mechanisms to generate effective test cases for ICP vulnerability detection. Our approach addresses key challenges in industrial protocol fuzzing by ensuring training stability through Wasserstein distance optimization, capturing long-range dependencies in protocol messages via self-attention, and maintaining test case diversity through gradient penalty enforcement.

Grenze ID: 01.GIJET.12.2.2_20

The remainder of this paper is organized as follows: Section 2 discusses related work in protocol fuzzing and deep learning applications. Section 3 provides necessary background on GANs and self-attention mechanisms. Section 4 details our proposed framework architecture. Section 5 presents experimental setup and results. Section 6 discusses implications and limitations. Finally, Section 7 con- cludes the paper and outlines future research directions.

II. Related Work

Fuzz testing has evolved significantly since its initial conception by Duran and Ntafos [2]. Early approaches focused on random input generation, but modern fuzzing techniques incorporate sophisticated program analysis to improve test effectiveness. Miller et al. pioneered the concept of fuzzing by demonstrating how random inputs could trigger unexpected behavior in software systems [3]. This foundational work inspired numerous subsequent developments in the field.

Protocol-specific fuzzing has received considerable attention due to the crit- ical nature of network communication in distributed systems. Chen et al. [4] developed IoTFuzzer, which leverages app-based analysis to discover memory corruption vulnerabilities in IoT devices. Their approach demonstrated the im- portance of understanding protocol semantics for effective vulnerability discov- ery. Similarly, Martin-Lopez et al. [5] addressed the challenge of inter-parameter dependencies in RESTful web APIs, highlighting the complexity of modern pro- tocol interactions.

The integration of machine learning with fuzzing represents a significant ad- vancement in automated vulnerability detection. Godefroid et al. [6] introduced Learn&Fuzz, which uses neural networks to learn input grammars for fuzzing. Their work demonstrated that machine learning could reduce reliance on manual specification analysis. Chen et al. [7] provided a comprehensive survey of fuzzing techniques, categorizing approaches based on their knowledge requirements and test generation strategies.

Generative Adversarial Networks have shown remarkable success in vari- ous domains, leading to their application in cybersecurity. Arjovsky et al. [8] proposed Wasserstein GANs, which address training instability through Earth- Mover distance optimization. This advancement made GANs more suitable for discrete sequence generation tasks like protocol fuzzing. Wu et al. [9] explored vulnerability detection with deep learning, demonstrating the potential of neural networks in security applications.

In the specific domain of industrial control systems, Li et al. [10] proposed an intelligent fuzzing data generation method based on deep adversarial learning. Their work established the feasibility of applying GANs to industrial protocol fuzzing but faced challenges with training stability and test case diversity. Our approach builds upon this foundation by incorporating self-attention mecha- nisms and improved training strategies to address these limitations.

Self-attention mechanisms, introduced by Vaswani et al. [11], have revolu- tionized sequence modeling by enabling parallel processing and better capture of long-range dependencies. These characteristics make self-attention particularly suitable for protocol message generation, where header fields often influence dis- tant parts of the message body. To our knowledge, our work represents the first integration of self-attention with GANs for industrial protocol fuzzing.

III. Background and Preliminaries

Industrial Control Protocols exhibit distinctive characteristics that differenti- ate them from general network protocols. ICPs are typically designed for real- time performance, resulting in short data frames, minimal overhead, and of- ten lack encryption mechanisms. These design choices prioritize operational ef- ficiency but can introduce security vulnerabilities. Common ICPs include Mod- bus, PROFINET, DNP3, and MQTT, each with unique message structures and communication patterns.

Fuzz testing, or fuzzing, involves automatically generating malformed inputs to test system robustness. Traditional fuzzing approaches can be categorized as black-box, white-box, or grey-box based on their knowledge of the target system. Black-box fuzzing, which requires no internal knowledge of the target, is particularly relevant for protocol fuzzing where source code may be unavailable. However, pure random fuzzing often proves inefficient for complex protocols with strict formatting requirements.

Generative Adversarial Networks consist of two neural networks—a generator and a discriminator—trained simultaneously through adversarial competition. The generator creates synthetic data samples, while the discriminator distin- guishes between real and generated samples. This adversarial process drives the generator

to produce increasingly realistic outputs. The original GAN formula- tion [12] suffers from training instability and mode collapse, where the generator produces limited varieties of samples.

Wasserstein GANs [8] address these limitations by using the Earth-Mover distance (Wasserstein-1) as the training objective. This distance metric provides more meaningful gradients than the Jensen-Shannon divergence used in stan- dard GANs. WGAN with Gradient Penalty (WGAN-GP) [13] further improves training stability by enforcing the Lipschitz constraint through gradient penalty rather than weight clipping.

The Wasserstein distance between two probability distributions P_r and P_g is defined as:

$$W(P_r, P_g) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathrm{E}_{(x,y)\sim\gamma}[\| x - y \|] \quad (1)$$

where $\Pi(P_r, P_g)$ denotes the set of all joint distributions with marginals Pr and Pg.

Self-attention mechanisms compute representations by weighting the impor- tance of different positions in the input sequence. For an input sequence $X = (x_1, x_2, ..., x_n)$, the self-attention operation produces output $Z = (z_1, z_2, ..., z_n)$ where each z_i is computed as a weighted sum of all input elements:

$$z_i = \sum_{j=1} a_{ij}(x_j W^V) \quad (2)$$

The attention weights a_{ij} are computed using a softmax function:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1} \exp(e_{ik})} \quad (3)$$

Where,

$$e_{ij} = \frac{(x_i W^{\vee})(x_j W^{\wedge})^{\iota}}{\sqrt{d_z}}$$

represents the compatibility between positions i and Multi-head self-attention extends this mechanism by performing multiple at- tention operations in parallel, allowing the model to jointly attend to information from different representation subspaces.

This capability is particularly valuable for protocol messages, where different header fields may influence various aspects of the message body.

IV. METHODOLOGY

HexGANFuzzer employs a structured approach to industrial protocol fuzzing, comprising four main components: data preprocessing, generator network, critic network, and training strategy. The overall framework architecture is illustrated in Figure 1.

A. Data Preprocessing

Industrial protocol messages typically use hexadecimal representation and ex- hibit variable lengths, presenting challenges for neural network processing. Our preprocessing pipeline addresses these issues through two main steps: message frame clustering and data conversion.

Message frame clustering groups messages by length to facilitate batch pro- cessing. Messages with identical lengths are grouped together, while groups con- taining insufficient samples (fewer than 5000 in our implementation) are merged with adjacent length groups.

This approach ensures that each training batch contains messages of uniform length, simplifying model architecture. After clus- tering, messages shorter than the maximum length in their group are padded with a special token 'u' to maintain consistent dimensions.

Data conversion transforms hexadecimal messages into numerical representa- tions suitable for neural network processing. We define a vocabulary comprising the 16 hexadecimal characters (0-9, a-f) plus the padding token 'u'. Each message is converted into a one-hot encoded vector $X \in R^{seq\ size \times 17}$, where seq size rep-
resents the maximum sequence length in the current group. This representation preserves the discrete nature of protocol messages while enabling gradient-based optimization.

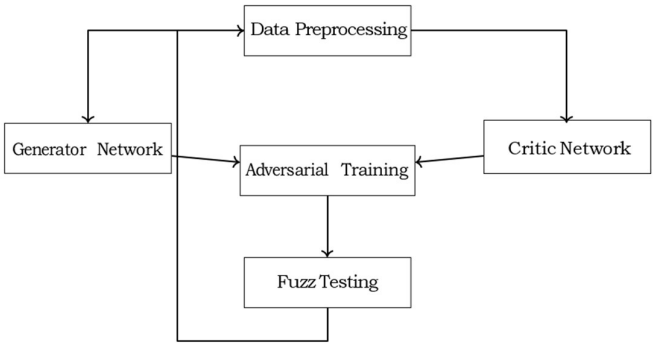

Fig. 1. HexGANFuzzer framework architecture

B. Generator Architecture

The generator network transforms random noise into synthetic protocol messages that resemble legitimate traffic. As shown in Figure 2, the generator comprises four main components: noise block, conversion block, transformer block, and output block. The noise block generates random Gaussian noise $z \in R^{1\times zd}$, where $zd = 50$ in our implementation. This stochastic input ensures diversity in generated messages across training iterations.

The conversion block transforms the noise vector into a structured repre-sentation through a fully connected layer followed by reshaping. This operation produces Noise Conversion Representation (NCR) $\tilde{z} \in R^{ss\times dm}$, where ss is the sequence length and $dm = 80$ is the feature dimension. NCR serves as a dif-

ferentiable alternative to word embeddings, addressing the challenge of discrete sampling in GAN-based sequence generation.

The transformer block employs multi-head self-attention to capture depen- dencies between different positions in the sequence. Our implementation uses four transformer layers with two attention heads each. Each layer includes resid- ual connections and layer normalization to facilitate training depth. The self- attention mechanism enables the model to learn how header fields influence subsequent message content, a crucial capability for generating valid protocol messages.

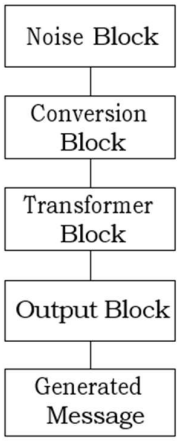

Fig. 2. Generator network architecture

The output block consists of a fully connected layer followed by softmax activation, producing a probability distribution over the vocabulary for each sequence position. The generator loss is defined as:

$$L_g = -E_{\tilde{x}\sim Pg}[D(\tilde{x})] \quad (4)$$

where D represents the critic function and P_g is the generator distribution.

C. Critic Architecture

The critic network evaluates the quality of generated messages by estimating the Wasserstein distance between real and synthetic distributions. Unlike tra- ditional discriminators that perform classification, the critic outputs a

scalar score representing sample quality. As illustrated in Figure 3, the critic comprises convolutional layers, self-attention layers, and a fully connected output layer.

The convolutional layers use varying kernel sizes (1 to 5) to capture local patterns at different scales. This design enables the model to learn protocol- specific structures, such as fixed-length headers that influence variable-length bodies. Each convolutional layer is followed by a self-attention layer that captures global dependencies across the sequence.

The critic loss incorporates the Wasserstein distance with gradient penalty:

$$L_c = \mathrm{E}_{\tilde{x}\sim \mathrm{P}_g}[D(\tilde{x})] - \mathrm{E}_{x\sim \mathrm{P}_r}[D(x)] + \lambda \mathrm{E}_{\hat{x}\sim \mathrm{P}_{\hat{x}}}[(\| \nabla_{\hat{x}} D(\hat{x}) \|_2 - 1)^2] \qquad (5)$$

where P_r and P_g represent real and generated distributions, $P_{\hat{x}}$ is the distribution of interpolated samples, and $\lambda = 10$ controls the gradient penalty strength.

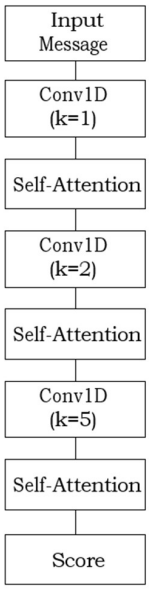

Fig. 3. Critic network architecture

D. Training Strategy

We employ an adversarial training strategy where the critic is updated multi- ple times per generator iteration. Specifically, we train the critic five times for each generator update to ensure accurate Wasserstein distance estimation be- fore generator optimization. This approach stabilizes training by preventing the generator from exploiting a poorly trained critic.

Model convergence is monitored using the empirical Wasserstein distance:

$$W\ distance = \mathrm{E}_{x\sim \mathrm{P}_r}[D(x)] - \mathrm{E}_{\tilde{x}\sim \mathrm{P}_g}[D(\tilde{x})] \qquad (6)$$

Training is considered stable when this metric shows consistent improvement over iterations.

To enhance testing comprehensiveness, we save model checkpoints every 10 epochs. This practice ensures diversity in generated test cases across different training stages. Additionally, messages that trigger anomalies during fuzzing are added to the training dataset after data augmentation, creating a feedback loop that improves vulnerability detection capability over time.

TABLE I. PERFORMANCE COMPARISON WITH DIFFERENT DATASET SIZES (10K/500K)

Models	Sensitivity	Specificity	Accuracy	F-measure
GPF	52.52%/57.20%	57.98%/57.30%	64.74%/65.10%	-/-
CNN-1D	84.37%/86.70%	97.43%/98.42%	86.26%/86.90%	67.01%/83.73%
LSTM	85.63%/82.03%	90.07%/98.56%	82.54%/87.00%	63.00%/83.88%
WGAN	76.96%/81.08%	91.15%/93.71%	81.65%/89.06%	79.60%/83.96%
HexGANFuzzer	82.74%/85.38%	97.75%/94.78%	86.08%/92.71%	82.08%/85.02%

V. EXPERIMENTAL EVALUATION

We conducted extensive experiments to evaluate HexGANFuzzer's performance on real industrial protocols. This section details our experimental setup, evalu- ation metrics, and comparative results against baseline methods.

A. Experimental Setup

Our evaluation focused on two widely used industrial protocols: MQTT (Mes- sage Queuing Telemetry Transport) and Modbus-TCP. MQTT is a lightweight publish-subscribe messaging protocol designed for constrained devices, while Modbus-TCP is a classic industrial protocol for supervisory control and data acquisition systems.

We implemented HexGANFuzzer using TensorFlow and conducted experi- ments on a machine with Intel i7-6700K CPU, 16GB RAM, and NVIDIA GTX 1080 Ti GPU. For MQTT testing, we used Mosquitto v1.5.8 as the broker and Paho-MQTT v1.0.2 as the client library. Modbus-TCP testing employed a widely used open-source implementation with simulated industrial devices.

Model hyperparameters were optimized through grid search. We set the batch size to 64, dropout rate to 0.3, and used Adam optimizer with learning rates of 0.0001 and 0.0004 for generator and critic respectively. The noise dimension zd was 50, feature dimension dm was 80, and we used 4 transformer layers with 2 attention heads each. Models were trained for 100 epochs with checkpoints saved every 10 epochs.

B. Evaluation Metrics

We developed comprehensive evaluation metrics to assess both machine learning performance and vulnerability detection capability:

F-measure Metrics: We adapted classification metrics to evaluate test case generation quality:

- Sensitivity: Percentage of correctly generated function codes
- Specificity: Percentage of correctly generated non-essential fields
- Accuracy: Overall correctness of generated messages
- F-measure: Harmonic mean of precision and sensitivity

Effectiveness of Vulnerability Detection (EVD): This composite met- ric evaluates testing depth and coverage through three sub-metrics:

- Acceptance Rate (ARTC): Percentage of accepted test cases
- Vulnerability Detection Ability (AVD): Percentage of test cases triggering anomalies
- Diversity of Generated Cases (DGC): Ratio of generated to training message categories

EVD is computed as the normalized sum of these sub-metrics.

Efficiency of Vulnerability Detection (EFVD): This metric assesses time efficiency through:

- Training Time (TT): Model training duration
- Time to Anomaly Trigger (TAT): Time until third anomaly detection EFVD is computed as the normalized inverse of these time metrics.

C. Comparative Analysis

We compared HexGANFuzzer against four baseline methods: General Purpose Fuzzer (GPF) [14], CNN-1D model, LSTM-based seq2seq model, and standard WGAN-based fuzzing. Each model was evaluated using 100,000 generated test cases on MQTT implementations.

Table 1 presents F-measure results on datasets of different sizes. HexGAN- Fuzzer achieved superior performance across most metrics, particularly for larger datasets. The accuracy improvement of 3.65% on the 500K dataset demonstrates our method's scalability to large-scale industrial environments.

Figure 4 illustrates EVD trends across training epochs. HexGANFuzzer con- sistently outperformed baseline methods, achieving approximately 8% higher EVD than standard WGAN-based fuzzing after 50 epochs. This improvement demonstrates the benefits of self-attention for capturing protocol semantics.

Efficiency analysis revealed that HexGANFuzzer achieved 19.45% higher EFVD than CNN-1D, 15.14% higher than LSTM, and 5.26% higher than stan- dard WGAN. This efficiency gain stems from parallel self-attention computation and stable training convergence.

D. Vulnerability Discovery

HexGANFuzzer successfully identified several critical vulnerabilities in tested protocols. In MQTT, we discovered a buffer exception vulnerability where mal- formed remaining length fields caused message parsing

errors. As illustrated in Figure 5, crafted messages could leave residual data in the broker's buffer, dis- rupting subsequent communications until buffer clearance.

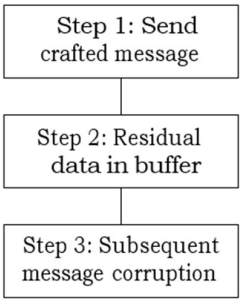

Fig. 5. Buffer exception vulnerability in MQTT

TABLE II. DETECTED ANOMALIES IN MODBUS-TCP

Anomaly Type	Frequency
Slave crash	29
Station off-line	164
Abnormal function code	207
Automatic window closure	13
Data length mismatch	190
Unknown attack pattern	197

For Modbus-TCP, Table 2 summarizes detected anomalies. HexGANFuzzer triggered various exceptions including slave crashes, station off-line events, and function code violations. The relatively short average time between anomaly trig- gers (0.52-28.47 minutes) demonstrates the method's effectiveness for practical security assessment.

VI. DISCUSSION

Our experimental results demonstrate that HexGANFuzzer effectively addresses key challenges in industrial protocol fuzzing. The integration of self-attention with WGAN-GP enables the model to learn complex protocol semantics without manual specification analysis, significantly reducing expertise requirements and testing preparation time.

The superior performance on larger datasets (500K samples) suggests that our method benefits from increased training data, making it suitable for indus- trial environments with extensive protocol traces. The stable training curves indicate that gradient penalty effectively enforces Lipschitz continuity, prevent- ing mode collapse and training divergence common in GAN-based approaches.

The discovered vulnerabilities, particularly the MQTT buffer exception, high- light real security risks in industrial communication. The ability to automatically generate test cases that trigger such subtle anomalies demonstrates the practical value of our approach for security assessment in critical infrastructure.

However, several limitations warrant consideration. First, the method re- quires substantial computational resources for training, which may be imprac- tical for resource-constrained industrial environments. Second, the clustering- based preprocessing may miss semantic relationships between messages of differ- ent lengths. Finally, the evaluation focused on two protocols; broader validation across diverse ICPs is needed.

Future work should address these limitations through model compression techniques [15], online learning for embedded deployment, and investigation of anti-random strategies to improve anomaly triggering probability. Additionally, extending the approach to encrypted industrial protocols represents an impor- tant research direction.

VII. CONCLUSION

This paper presented HexGANFuzzer, a novel fuzzing framework that combines WGAN-GP with self-attention mechanisms for industrial protocol security as- sessment. Our approach generates diverse, realistic test cases that effectively trigger vulnerabilities while maintaining protocol compliance. Extensive eval- uations demonstrated superior performance compared to traditional and deep learning-based fuzzing methods.

The key innovations of our work include: (1) stable adversarial training through Wasserstein distance optimization and gradient penalty; (2) effective protocol semantics capture via self-attention mechanisms; (3) comprehensive evaluation metrics for fuzzing effectiveness and efficiency; and (4) demonstrated vulnerability discovery in real industrial protocols.

HexGANFuzzer represents a significant step toward automated, intelligent se- curity assessment for industrial control systems. By reducing reliance on manual specification analysis and expertise, our method makes comprehensive vulnera- bility testing more accessible for industrial operators. As critical infrastructure faces increasing cyber threats, such automated security assessment tools become essential for maintaining operational safety and reliability.

REFERENCES

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," Business & Information Systems Engineering, vol. 6, no. 4, pp. 239–242, 2014.

[2] J. W. Duran and S. C. Ntafos, "An evaluation of random testing," IEEE transac- tions on Software Engineering, vol. SE-10, no. 4, pp. 438–444, 1984.

[3] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of unix utilities," in Communications of the ACM, vol. 33, no. 12, 1990, pp. 32–44.

[4] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang, "Iotfuzzer: Discovering memory corruptions in iot through app- based fuzzing," in NDSS, 2018.

[5] A. Martin-Lopez, S. Segura, and A. Ruiz-Cortes, "A catalogue of inter-parameter dependencies in restful web apis," in International Conference on Service-Oriented Computing. Springer, 2019, pp. 399–414.

[6] P. Godefroid, H. Peleg, and R. Singh, "Learn&fuzz: Machine learning for input fuzzing," in 2017 32nd IEEE/ACM International Conference on Automated Soft- ware Engineering (ASE). IEEE, 2017, pp. 50–59.

[7] C. Chen, B. Cui, J. Ma, R. Wu, J. Guo, and W. Liu, "A systematic review of fuzzing techniques," Computers & Security, vol. 75, pp. 118–137, 2018.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," arXiv preprint arXiv:1701.07875, 2017.

[9] F. Wu, J. Wang, J. Liu, and W. Wang, "Vulnerability detection with deep learn- ing," in 2017 3rd IEEE International Conference on Computer and Communica- tions (ICCC). IEEE, 2017, pp. 1298–1302.

[10] Z. Li, H. Zhao, J. Shi, Y. Huang, and J. Xiong, "An intelligent fuzzing data gener- ation method based on deep adversarial learning," IEEE Access, vol. 7, pp. 49 327– 49 340, 2019.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L- . Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, 2014, pp. 2672–2680.

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Im- proved training of wasserstein gans," in Advances in neural information processing systems, 2017, pp. 5767–5777.

[14] J. DeMott, R. Enbody, and W. F. Punch, "Revolutionizing the field of grey-box attack surface testing with evolutionary fuzzing," in BlackHat and Defcon, 2007.

[15] M. Li, J. Lin, Y. Ding, Z. Liu, J.-Y. Zhu, and S. Han, "Gan compression: Efficient architectures for interactive conditional gans," arXiv preprint arXiv:2003.08936, 2020.