MagDDAE: Integrating Manifold Learning and Diffusion Purification for Three-Lane API Security

Devina Tirza Nugroho
Cyber Security Program
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
devina.nugroho@binus.ac.id

Frederick Yonathan Ehowu Mendrofa
Cyber Security Program
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
frederick.mendrofa@binus.ac.id

Frederick Benjamin Widiya
Cyber Security Program
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
frederick.widiya@binus.ac.id

Yohan Muliono
Cyber Security Program
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
ymuliono@binus.edu

***Abstract*—Machine Learning-based Web Application Firewalls (WAFs) face a critical paradox: robust defense against adversarial evasion attacks typically comes at the cost of prohibitive computational latency. This paper resolves this Efficiency-Resilience Gap with MagDDAE, a hybrid framework integrating manifold learning with a novel One-Step diffusion-based purifier. Governed by an adaptive Three-Lane Traffic Handling architecture, the system utilizes a lightweight manifold detector to dynamically route traffic. Instead of purifying all inputs, MagDDAE steers 90.6% of traffic through the Express Lane and routes only ambiguous samples (7.76%) to the purification lane. Experiments on the ATRDF dataset demonstrate robust defense, achieving an F1-Score of 0.9655 and reducing successful security breaches by 35.3% compared to standard WAFs. Crucially, by preemptively discarding gross anomalies (Lane 3), the system achieved a net 1.15% reduction in computational cost alongside a 65.6% reduction in False Alarms, validating its operational feasibility for real-time, high-throughput API security.**

***Index Terms*—Web Application Firewall (WAF), Adversarial Attacks, Machine Learning, Diffusion Models, Manifold Learning, API Security, Three-Lane Architecture.**

I. Introduction

Application Programming Interfaces (APIs) serve as the foundational architecture for modern data exchange [1] [2], yet their ubiquity necessitates robust security to sustain rapid innovation [3]. As legacy signature-based WAFs proved ineffective against zero-day and obfuscated threats [4], the industry shifted toward intelligent machine learning (ML) models [5] [6]. However, this transition introduces a critical vulnerability: **Adversarial Attacks**. By making minute modifications to API parameters, attackers can execute malicious payloads while evading ML-based detection [7] [8] [9].

Recent findings by Holla et al. (2025) underscore this risk, demonstrating that adversarial techniques such as FGSM, PGD, and DeepFool can degrade Intrusion Detection System (IDS) accuracy from 85% to as low as 50% [10]. This creates a defensive paradox: more sophisticated ML models inherently expand the attack surface for evasion. While adversarial training and SHAP feature selection offer partial mitigation, Holla et al. conclude that high-accuracy protection requires the integration of detection-based defenses, such as MagNet, to complement traditional training-based methods [10].

Currently, there are two previous studies that cover this adversarial problem:

1) Manifold-based Autoencoders (MagNet) [11]: This method uses autoencoders to learn the manifold of normal traffic, then flags an anomaly by the reconstruction error. However, we found that they often fail to capture the fine details of API traffic (over-smoothing), leading to poor detection of stealthy attacks.
2) Diffusion-Scheduled Denoising Autoencoders (DDAE) [15]: This approach adds noise to the input, then iteratively removes it, purifying the input back into the valid data manifold. However, the standard iterative process is computationally inefficient.

This creates a trade-off: we can have a fast WAF that is easily broken, or a secure WAF that is inefficient.

To solve this problem, **MagDDAE**, a hybrid framework that orchestrates the efficiency of MagNet with the robustness of DDAE via a **Three-Lane Traffic Handling** architecture, was proposed. In this system, every incoming API request is first analyzed by the MagNet manifold detector. MagNet then calculates the reconstruction error, which serves as a reference for routing the request. Requests with low errors (Green/Express) will bypass strict defenses and go directly to the WAF; requests with moderate errors (Yellow/Purification) are marked as suspicious and sent to the DDAE model for purification before classification; and requests with very high errors (Red/Block) are marked as malicious and immediately blocked to conserve resources.

The main contributions of this paper are:

- **MagDDAE Framework:** A novel architecture that leverages MagNet as a fast gatekeeper and DDAE as a robust purifier with the implementation of the **"One-Step Purification"** mechanism. This method has been shown to reduce successful security breaches by 35.3% compared to a standard WAF.
- **Three-Pass Mechanism:** To balance security and speed, a new traffic handling mechanism is introduced. By selectively purifying suspicious traffic, the system reduces false alarms by 65.6% compared to all traffic passing through DDAE while achieving a net reduction in effective computational cost of 1.15% compared to a standard WAF setup.
- **Constraint-Based Attack Generator:** Addressing the validity challenges highlighted by Grini et al. [16], a constraint-based generator has been implemented. Unlike standard gradient methods, which frequently violate domain constraints, this mechanism projects perturbations onto a valid, suitable set, ensuring that all simulated attacks maintain semantic integrity and represent functional, executable API payloads.

II. Related Works

A. ML-based WAF and Adversarial Attacks

The limitations of signature-based Web Application Firewalls (WAFs) have led to the adoption of Machine Learning (ML) and Deep Neural Networks (DNN), which offer better adaptability in learning complex attacks directly from traffic. Recent research [17] [18] confirms that DNN-based architectures significantly outperform classical baselines, achieving high accuracy and recall even on sophisticated vectors such as command injection. However, despite their accuracy, ML-based classifiers are vulnerable to adversarial attacks, compromising the security of these supposedly robust systems [24].

Recent research by Holla et al. (2025) explicitly evaluated these threats on Cloud IDS. They demonstrated that adversarial attacks can degrade detection accuracy from 85% to as low as 50% [10].

B. Defensive Mechanisms

*1) **Adversarial Training**:* Adversarial training remains an important defense paradigm [10]–[12]. It augments the training dataset with adversarial examples, forcing the model to learn robust decision boundaries. However, this approach often fails to generalize to unseen attack patterns or "Blind-Spot" inputs residing in low-density manifolds [13], [14]. Furthermore, the significant computational overhead required to retrain models against zero-day threats hinders adaptability in real-time environments.

*2) **Autoencoder-based Defenses (MagNet)**:* MagNet [10] defends against adversarial attacks by learning the manifold of legitimate traffic using autoencoders. During inference, it calculates the "Reconstruction Error" (MSE) between the input and the autoencoder's output. If the error exceeds a learned threshold, the input is rejected as adversarial. While effective for images, applying this to sparse API payloads often triggers a "latent bottleneck." The compression required to map high-dimensional inputs to a lower-dimensional manifold results in *over-smoothing*, causing fine-grained adversarial perturbations to be lost during decoding and effectively masking stealthy attacks [15]. As established by Khan et al. [23], the fundamental vulnerability of autoencoder-based detectors lies in their high generalization capability. This allows the model to inadvertently reconstruct malicious samples with minimal error, thereby concealing adversarial noise within the benign data manifold and bypassing the detection threshold.

*3) **Diffusion Models (DDAE)**:* Sattarov et al. (2025) introduced the Diffusion-Scheduled Denoising Autoencoder (DDAE), which injects controlled noise to enhance feature separation by gradually modifying input samples. [15]. DDAE utilizes a forward diffusion process to gradually inject scheduled noise and a reverse generative process to reconstruct the original input from the noisy state. This effectively "purifies" the sample by projecting it back into the valid data manifold. However, the standard implementation relies on an iterative Markov Chain for reconstruction, which is computationally prohibitive for real-time WAFs.

C. The Efficiency-Robustness Gap

Despite the "One-Step" optimization, diffusion-based purification remains computationally heavier than standard inference. Applying purification to 100% of incoming API traffic would introduce unnecessary latency for the vast majority of legitimate users. While relying solely on lightweight autoencoders (MagNet) compromises security against stealthy attacks due to over-smoothing. To address this trade-off, the MagDDAE framework is proposed.

III. Methodology

A. System Architecture and Workflow

MagDDAE utilizes a single forward pass for both tasks, unlike previous systems that require separate models for detection and defense. It combines the architecture of MagNet and DDAE, encapsulated in a dynamic mechanism called **Three-Lane Traffic Handling**, as illustrated in Figure 1.

1) Phase 1: Input and Pre-processing: The pipeline begins with the ingestion of raw API requests. Each request is represented as a vector of 26 features, which undergoes normalization via a Standard Scaler to produce the normalized feature set $\mathbf{x}$. This standardization ensures consistent input distribution for the downstream neural networks.

2) Phase 2: MagNet Gatekeeper: The normalized features are fed into the **MagNet Autoencoder**, which functions as the system's "Gatekeeper." For each input traffic batch $X \in \mathbb{R}^{B\times D}$, we define a fixed time step vector $\mathbf{t} = \mathbf{1} \in \mathbb{R}^{B}$. Treating the input batch as a noisy state x_t, the model performs **One-Step Reconstruction** to predict the clean origin X_{recon}:

$$X_{recon} = \text{MagNet}(X, \mathbf{t}) \tag{1}$$

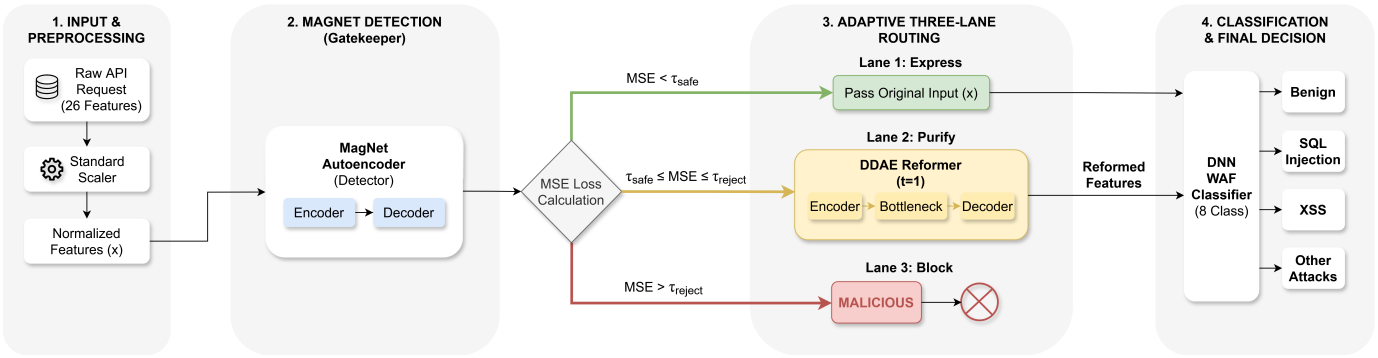

Fig. 1. **The MagDDAE Framework.** Traffic is dynamically routed based on Reconstruction Error (MSE) relative to learned thresholds: τ_{safe} (95th percentile of benign error) allows traffic to bypass purification via the Express Lane, while τ_{reject} (99th percentile) immediately blocks gross anomalies. Ambiguous samples fall between these thresholds and undergo **One-Step Purification** ($t = 1$) before WAF classification.

3) Phase 3: Adaptive Three-Lane Routing: During inference, the system calculates the Reconstruction Error vector MSE:

$$MSE = \frac{1}{D}\sum_{i=1}^{D}(X - X_{recon})^2 \tag{2}$$

Based on this error, the system dynamically routes traffic. We define two decision thresholds, τ_{safe} and τ_{reject}, which are determined empirically based on the 95th and 99th percentiles of the reconstruction error on a benign validation set. The routing logic is formalized in **Algorithm 1**.

- **Lane 1 (Green - Express).** Traffic with $MSE < \tau_{safe}$ is considered similar to the benign manifold. The system immediately passes the input to the WAF classifier to maintain high-quality data accuracy, avoiding potential data integrity issues from the purification step.
- **Lane 2 (Yellow - Purify).** Inputs falling within the uncertainty interval ($\tau_{safe} \leq MSE \leq \tau_{reject}$) trigger the **DDAE Reformer**. The system activates the diffusion process with a fixed time step ($t = 1$) to purify the adversarial perturbations, projecting the sample back onto the benign manifold before forwarding the reformed features ($\hat{x}_{ddae}$) to the classifier.
- **Lane 3 (Red - Block).** Inputs exhibiting excessive deviation ($MSE > \tau_{reject}$) are classified as high-confidence malicious. These requests are blocked immediately at the gateway for computational efficiency.

4) Phase 4: Classification and Final Decision: The final stage employs a Deep Neural Network (DNN) WAF Classifier. Depending on the routing decision, the WAF receives either the original features (from Lane 1) or the reformed features (from Lane 2). It outputs an 8-class prediction, categorizing the traffic as Benign or identifying specific attack vectors.

Algorithm 1 MagDDAE Vectorized Three-Lane Inference

Require: Batch inputs X, Thresholds $\tau_{safe}, \tau_{reject}$, WAF Model f_{waf}

Step 1: MagNet Detection (Gatekeeper)
$X_{mag} \leftarrow$ MagNet(X) {Fast Autoencoder Reconstruction}
$MSE \leftarrow \text{mean}((X_{mag} - X)^2, \text{dim} = 1)$ {Calculate Reconstruction Error}
Step 2: Lane Allocation (Masking)
$M_{green} \leftarrow MSE < \tau_{safe}$ {Lane 1: Trusted}
$M_{red} \leftarrow MSE > \tau_{reject}$ {Lane 3: Anomalous}
$M_{yellow} \leftarrow \neg M_{green} \wedge \neg M_{red}$ {Lane 2: Suspicious}
Step 3: Conditional Execution
$Y_{pred} \leftarrow \text{zeros}(X.\text{size}(0))$
if M_{green}.any() **then**
  $Y_{pred}[M_{green}] \leftarrow f_{waf}(X[M_{green}])$ {Pass Original Data}
end if
if M_{yellow}.any() **then**
  $t \leftarrow \text{ones}(M_{yellow}.\text{sum}())$ {Set Timestep $t = 1$}
  $X_{clean} \leftarrow \text{DDAE}(X[M_{yellow}], t)$ {Activate Diffusion Reformer}
  $Y_{pred}[M_{yellow}] \leftarrow f_{waf}(X_{clean})$ {Pass Reformed Data}
end if
Step 4: Blocking
$Blocked \leftarrow M_{red}$ {Flag Lane 3 as Malicious}
return $Y_{pred}, Blocked$

B. Dataset Description

The dataset used in this study is the **API Traffic Research Dataset Framework (ATRDF)** [19], which consists of 182,767 total samples, 143,431 benign requests and 39,336 malicious requests. ATRDF provides detailed **multiclass labels** of seven specific attack types: SQL Injection, XSS, RCE, Log Forging, Cookie Injection, Directory Traversal, and LOG4J. This allows us to directly address the limitation noted by Holla et al. [10], who identified the lack of multiclass adversarial evaluation as a critical gap in current research.

C. Threat Model and Attack Generation

The system was evaluated against three gradient-based evasion attacks:

- **FGSM (Fast Gradient Sign Method):** A single-step attack that perturbs inputs in the direction of the loss

gradient to maximize error [20].

- **PGD (Projected Gradient Descent):** An iterative variant of FGSM which applies multiple small updates to refine adversarial noise while projecting perturbations back into a valid ϵ-ball [21].
- **DeepFool:** An optimization-based attack seeking the minimal perturbation required to cross the closest decision boundary, often producing highly stealthy samples [22].

These attacks were generated using the *Constraint-Based Generator* adapted from Grini et al. [16] to ensure attack validity. Unlike their filtering approach, perturbation onto a valid manifold was projected to ensure no data loss.

The feature space was partitioned into three semantic categories: Binary (F_{bin}), Count (F_{count}), and One-Hot Groups (G). Scalar features are constrained via clipping and rounding:

$$x_i = \begin{cases} \lfloor \text{clip}(\tilde{x}_i, 0, 1) \rceil & \text{if } i \in F_{bin} \\ \lfloor \max(0, \tilde{x}_i) \rceil & \text{if } i \in F_{count} \end{cases} \tag{3}$$

For categorical groups (e.g., HTTP Method), we enforce mutual exclusivity using an $\arg\max$ operator. This ensures that for every group $g \in G$, exactly one state is active:

$$x_j = \begin{cases} 1 & \text{if } j = \arg\max_{k \in g}(\tilde{x}_k) \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in g \tag{4}$$

This projection ensures that all generated adversarial samples represent executable and semantically valid API payloads.

D. Target WAF Implementation

The target model is a Deep Neural Network (DNN) with three hidden layers, batch normalization, and dropout regularization to mitigate overfitting. The model was trained for 50 epochs using the Adam optimizer with an initial learning rate of 0.001. On clean validation data, the model achieves a baseline accuracy of 98.72%, serving as a high-standard benchmark for evaluating adversarial robustness.

E. Experimental Environment

All models were implemented in PyTorch and evaluated on an NVIDIA T4 GPU (12GB RAM). Inference latency is counted as the average wall-clock time per batch ($B = 512$) across the entire test set to simulate high-throughput conditions.

F. Evaluation Scenario: "All-in-One" Test Set

To simulate a real-world environment where a WAF faces a heterogeneous mix of traffic, an "All-in-One" test set was conducted with this distribution:

- **60% Benign Traffic:** Normal API user requests.
- **20% Malicious Traffic:** Standard known attacks.
- **10% Adversarial Traffic:** FGSM, PGD, and DeepFool attacks.
- **10% Fuzzed Traffic:** Attack generated using the semantic-preserving fuzzing logic like in the WAF-A-MOLE [9].

[1]https://github.com/Trevillie/MagNet

[2]https://github.com/sattarov/AnoDDAE

G. Performance Metrics

We evaluate detection performance using standard classification metrics: Accuracy, Precision, Recall, and F1-Score. Additionally, to quantify the operational impact of the Three-Lane architecture, we define three custom metrics:

*1) **Breach Prevention Rate (BPR)**:* This metric measures the relative reduction in successful adversarial evasions (False Negatives) compared to the baseline defense (DDAE). It measures the improvement in security coverage:

$$BPR = \left(\frac{FN_{baseline} - FN_{ours}}{FN_{baseline}} \right) \times 100\% \tag{5}$$

where FN represents the count of malicious samples misclassified as benign.

*2) **False Alarm Reduction (FAR)**:* To assess the reduction in operational burden ("Alert Fatigue"), we measure the decrease in False Positives (benign samples misclassified as malicious):

$$FAR = \left(\frac{FP_{baseline} - FP_{ours}}{FP_{baseline}} \right) \times 100\% \tag{6}$$

*3) **Effective Operational Cost (C_{eff})**:* To measure the efficiency gain from dynamic flow, we calculate the expected computational cost per request. Let C_{waf}, C_{mag}, and C_{purify} represent the isolated computational costs (FLOPs) for the WAF classifier, the MagNet gatekeeper, and the diffusion reformer, respectively.

The effective cost is the probability-weighted sum of the operations performed in each lane:

$$\begin{aligned} C_{eff} = {} & P_{green}(C_{mag} + C_{waf}) \\ & + P_{yellow}(C_{mag} + C_{purify} + C_{waf}) \\ & + P_{red} C_{mag} \end{aligned} \tag{7}$$

where P_L is the fraction of traffic routed to Lane L. Crucially, for Lane 3 (Red), the cost is strictly limited to C_{mag}, as the sample is dropped before WAF execution.

IV. Results and Analysis

A. Attack Validity Verification

We evaluated the validity of samples generated by FGSM, PGD, and DeepFool using the constraint logic defined in Section III.

TABLE I
ADVERSARIAL ATTACK VALIDITY ANALYSIS

Attack Method	Constraint	Semantic	Combined
FGSM	55.8%	49.7%	27.7%
PGD	59.3%	49.7%	26.5%
DeepFool	**80.2%**	**49.7%**	**39.8%**
CW	62.7%	49.7%	33.6%

As shown in Table I, DeepFool achieved the highest constraint validity (80.2%), higher than Carlini & Wagner (CW), which has the highest proportion of valid adversarial samples reported in the Grini et al. paper. This ensures that our defense is tested against functionally plausible attacks.

B. Overall Detection Performance

We conducted our primary evaluation on the "All-in-One" test chamber, representing a realistic mix of Benign (60%), Malicious (20%), Adversarial (10%), and Fuzzed (10%) traffic.

TABLE II
DEFENSE PERFORMANCE ON MIXED API TRAFFIC

Model	Accuracy	Precision	Recall	F1-Score
Plain WAF	0.9691	0.9999	0.9229	0.9598
MagNet	0.7631	0.6940	0.7294	0.7113
Base DDAE	0.9498	0.9468	0.9267	0.9366
MagDDAE	**0.9729**	**0.9815**	**0.9501**	**0.9655**

MagDDAE outperformed all baselines. Notably, it achieved a Recall of **95.01%**, significantly higher than the standalone DDAE (92.67%). This confirms that using MagNet as a gatekeeper enhances precision by filtering benign traffic before it reaches the purifying step.

C. Robustness by Attack Vector

To understand specific strengths, we decomposed the evaluation into distinct attack vectors. Table III details the Detection Rate (Recall) for each scenario.

TABLE III
DETECTION RATE (RECALL) VS. SPECIFIC ATTACK VECTORS

Attack Type	Plain WAF	MagNet	Base DDAE	MagDDAE
FGSM	84.19%	74.33%	91.43%	**94.00%**
PGD	84.19%	74.25%	91.43%	**94.00%**
DeepFool	84.19%	73.79%	91.43%	**94.03%**

Against **DeepFool**—the most valid and stealthy attack, MagDDAE improved detection from 84.19% to **94.03%**, demonstrating the effectiveness of the One-Step Purification process.

D. Multiclass Classification Performance

We evaluated the ability to classify specific attack types on the mixed test set.

TABLE IV
MULTICLASS PERFORMANCE

Model	Accuracy	Precision	Recall	F1-Score
Plain WAF	0.9654	0.9650	0.9654	0.9608
MagNet	0.7162	0.7322	0.7162	0.7200
Base DDAE	0.9348	0.9211	0.9348	0.9257
MagDDAE	**0.9661**	**0.9663**	**0.9661**	**0.9618**

MagDDAE achieved the highest multiclass performance (F1 0.9618), effectively recovering the classification capability of the baseline WAF (98.72%). This confirms that the purification process preserves the fine-grained features necessary for accurate attack attribution.

It is important to note that the system's feature reformatting is used solely for analysis and classification. The backend will still process the raw requests to preserve the original API request.

E. Effective Computational Efficiency

We analyzed the computational cost (FLOPs) across different processing paths to quantify the operational impact of the Three-Lane architecture.

TABLE V
EFFECTIVE COMPUTATIONAL EFFICIENCY (PER SAMPLE)

Method	Min (Lane 3)	Max (Lane 2)	Average	Static
Plain WAF	359.94K	359.94K	359.94K	359.94K
MagNet	360.77K	360.77K	360.77K	360.77K
Base DDAE	369.54K	369.54K	369.54K	369.54K
MagDDAE	**936**	**369.54K**	**355.78K**	369.54K

As shown in Table V, MagDDAE introduces a dynamic cost structure:

- **Lane 3 (Minimum):** Gross anomalies (1.60% of traffic) are blocked immediately, costing only **936 FLOPs** (a 99.7% reduction vs WAF).
- **Lane 2 (Maximum):** Ambiguous traffic (7.76%) undergoes full purification, costing 369.54K FLOPs.
- **Lane 1 (Standard):** Safe traffic (90.64%) bypasses purification, incurring only standard detection costs.

Reason for Cost Reduction: This intelligent routing yields an **Effective Average Cost of 355.78K FLOPs—1.15% lower** than the baseline WAF. This net gain occurs because the substantial savings from preemptively discarding Lane 3 anomalies (skipping the heavy WAF execution) effectively subsidize the lightweight monitoring of valid traffic. Specifically, while Lane 2 introduces a purification overhead, it is mathematically outweighed by the 99.7% cost reduction for every sample routed to Lane 3. Consequently, the architecture turns the detection of gross anomalies into a latency-optimization mechanism rather than a computational burden.

F. Security Impact: Breach Prevention

We analyzed the absolute number of successful evasions (False Negatives) in our test sample compared to the standard WAF:

- **Plain WAF:** Missed 607 attacks.
- **MagDDAE:** Missed 393 attacks.

MagDDAE prevented **214 additional breaches**, representing a **35.3% reduction** in security incidents compared to the baseline WAF.

While the branching architecture increases the average inference time ($0.299s$) compared to the standalone DDAE ($0.090s$), this trade-off is justified by the massive reduction in operational noise.

- **DDAE False Alarms:** 410 legitimate requests blocked.
- **MagDDAE False Alarms:** 141 legitimate requests blocked.

This constitutes a **65.6% improvement** in False Alarm reduction. In high-security contexts, the "alert fatigue" caused by false positives vastly outweighs the millisecond-scale latency increase.

V. Conclusion

This research addresses the vulnerability of modern API-based WAFs to adversarial evasion by bridging the Efficiency-Resilience Gap between lightweight manifold detectors and robust generative purifiers. The proposed **MagDDAE** framework utilizes a **Three-Path Traffic Handling** architecture to optimize security and performance simultaneously. Empirical results confirm a peak **F1 score of 0.9655** and a **35.3%** reduction in successful breaches. Crucially, MagDDAE eliminates the traditional security-latency trade-off by achieving a **1.15%** reduction in computational costs alongside a **65.6%** decrease in false alarms compared to standard WAFs.

Future research will focus on two primary enhancements to improve the framework's performance and scalability. First, efforts will be directed toward refining multiclass attribution accuracy (currently 84%) to align with our high standards for binary detection. This involves optimizing the latent representation to better distinguish between specific attack types, thereby improving forensic analysis and incident response. Second, the architecture will be integrated with **Federated Learning (FL)** to create a collaborative, privacy-preserving defense network. By allowing distributed edge gateways to share learned model updates rather than raw traffic logs, the system can adapt to global zero-day threats in real-time without compromising data privacy. This evolution will transform MagDDAE from a localized defense into a globally distributed, self-evolving security ecosystem.

Acknowledgements

We would like to thank Cisco and Ariel University for providing the dataset and making the API Security Challenge data publicly available via GitHub. This resource was instrumental in the evaluation of our proposed models.

References

[1] N. Melville and R. Kohli, "Roadblocks to Implementing Modern Digital Infrastructure: Exploratory Study of API Deployment in Large Organizations," Proceedings of the Annual Hawaii International Conference on System Sciences, Jan. 2021, doi: 10.24251/HICSS.2021.723.

[2] B. Shivnani, V. Shrivastava, A. Pandey, and A. K. Tiwari, "The Evolution of APIs: From Basic Interfaces to Intelligent Systems," International Journal of Research Publication and Reviews, vol. 6, no. 4, pp. 5177–5181, Apr. 2025, [Online]. Available: https://ijrpr.com/uploads/V6ISSUE4/IJRPR42126.pdf

[3] OWASP API Security Project — OWASP Foundation. https://owasp.org/www-project-api-security/ (accessed Nov. 10, 2025).

[4] S. Applebaum, T. Gaber, and A. Ahmed, "Signature-based and Machine-Learning-based Web Application Firewalls: A short survey," Procedia Computer Science, vol. 189, pp. 359–367, Jan. 2021, doi: 10.1016/j.procs.2021.05.105.

[5] R. Z. Muttaqin and D. Sudiana, "Design of realtime web application firewall on Deep Learning-Based to improve Web application security," Jurnal Penelitian Pendidikan IPA, vol. 10, no. 12, pp. 11121–11129, Jan. 2025, doi: 10.29303/jppipa.v10i12.8346.

[6] I. Debicha, T. Debatty, J.-M. Dricot, and W. Mees, "Adversarial Training for Deep Learning-based Intrusion Detection Systems". Portugal: ICONS 2021, The Sixteenth International Conference on Systems, 2021.

[7] K. He, D. D. Kim and M. R. Asghar, "Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey," in IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 538-566, Firstquarter 2023, doi: 10.1109/COMST.2022.3233793.

[8] E. Alshahrani, D. Alghazzawi, R. Alotaibi, and O. Rabie, "Adversarial attacks against supervised machine learning based network intrusion detection systems," PLoS ONE, vol. 17, no. 10, p. e0275971, Oct. 2022, doi: 10.1371/journal.pone.0275971.

[9] L. Demetrio, A. Valenza, G. Costa, and G. Lagorio, "WAF-A-MoLE: Evading Web Application Firewalls through Adversarial Machine Learning," Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC '20), Association for Computing Machinery, New York, NY, USA, pp. 1745–1752, 2020. doi: 10.1145/3341105.3373962.

[10] H. Holla, S. R. Polepalli, and A. A. Sasikumar, "Adversarial Threats to Cloud IDS: Robust defense with adversarial training and feature selection," IEEE Access, vol. 13, pp. 84992–85003, Jan. 2025, doi: 10.1109/access.2025.3567038.

[11] D. Meng and H. Chen, "MagNet: A Two-Pronged Defense against Adversarial Examples," CCS '17: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 135–147, Oct. 2017, doi: 10.1145/3133956.3134057.

[12] M. S. Haroon and H. M. Ali, "Adversarial training against adversarial attacks for Machine Learning-Based Intrusion Detection systems," Computers, Materials & Continua/Computers, Materials & Continua (Print), vol. 73, no. 2, pp. 3513–3527, Jan. 2022, doi: 10.32604/cmc.2022.029858.

[13] F. Marchiori, M. Alecci, L. Pajola, and M. Conti, "DUMB and DUMBer: Is adversarial training worth it in the real world?" in Computer Security – ESORICS 2025, 2025, pp. 228–248, doi: 10.1007/978-3-032-07884-1_12.

[14] H. Zhang, H. Chen, Z. Song, D. S. Boning, I. S. Dhillon, and C.-J. Hsieh, "The limitations of adversarial training and the blind-spot attack," in Proc. 7th Int. Conf. Learn. Representations (ICLR), New Orleans, LA, USA, May 2019. [Online]. Available: https://openreview.net/forum?id=HylTBhA5tQ

[15] T. Sattarov, M. Schreyer, and D. Borth, "Diffusion-Scheduled Denoising Autoencoders for Anomaly Detection in Tabular Data," Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), pp. 2478–2489, Aug. 2025, doi: 10.1145/3711896.3736910.

[16] A. Grini, O. Taheri, B. E. Khamlichi, and A. E. Fallah-Seghrouchni, "Constrained Network Adversarial Attacks: Validity, Robustness, and Transferability," 2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), pp. 771–776, Jun. 2025, doi: 10.1109/dcoss-iot65416.2025.00117.

[17] B. Dawadi, B. Adhikari, and D. Srivastava, "Deep Learning Technique-Enabled Web Application firewall for the detection of web attacks," Sensors, vol. 23, no. 4, p. 2073, Feb. 2023, doi: 10.3390/s23042073.

[18] X. Wang, J. Zhai, and H. Yang, "Detecting command injection attacks in web applications based on novel deep learning methods," Scientific Reports, vol. 14, no. 1, p. 25487, Oct. 2024, doi: 10.1038/s41598-024-74350-3.

[19] Lavian, Shmuel, and Ariel University, Ariel Cyber Innovation Center (ACIC), "The API Traffic Research Dataset Framework (ATRDF)." Jan. 2023. [Online]. Available: https://github.com/ArielCyber/Cisco_Ariel_Uni_API_security_challenge

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv (Cornell University), Dec. 2014, doi: 10.48550/arxiv.1412.6572.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," arXiv (Cornell University), Jun. 2017, doi: 10.48550/arxiv.1706.06083.

[22] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks," IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582, Jun. 2016, doi: 10.1109/cvpr.2016.282.

[23] W. Khan, N. Ebrahim, M. Ishrat, H. Alkahtani, and T. Aldhyani, "An adversarial variational graph autoencoder with contrastive learning for robust anomaly detection in large scale attributed networks," Journal of Big Data, Dec. 2025, doi: 10.1186/s40537-025-01342-z.

[24] S. Sultana and A. Rahman, "A Multi-Layered defense framework against adversarial attacks on ML-based web application firewalls," Open Access Research Journal of Science and Technology, vol. 15, no. 2, pp. 063–078, Nov. 2025, doi: 10.53022/oarjst.2025.15.2.0137.