Executive Summary

of

the thesis entitled

“Technique for Proactive Detection of Vulnerabilities in Enterprise Web Application”

Submitted By

Dr. Bela Shah (FOTE/1064)

Under the guidance of

Dr. Apurva Shah

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING FACULTY OF TECHNOLOGY AND ENGINEERING

THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA

VADODARA 390 001

February 2026

TABLE OF CONTENT OF THE EXECUTIVE SUMMARY

TABLE OF CONTENT OF THESIS..... ii
INTRODUCTION..... 1
Problem Description and Research Gap..... 2
Motivation..... 3
Problem Statement..... 3
Objectives..... 3
Scope..... 4
Research Methodology For Work done..... 4
CONCLUSION.....5
Outcome of Research Work.....7
Research Contribution.....8
FUTURE WORK.....9
BIBLIOGRAPHY.....10

Table of Contents of Thesis

CHAPTER 1: INTRODUCTION **8**
1.1 Background and Evolution of Cloud-Based Web Applications 8
1.2 Taxonomy of Web Application Vulnerabilities and Attacks 9
1.2.1 Web Application Vulnerabilities 9
1.2.2 Attack Methodologies and Patterns 12
1.3 Limitations of Existing Intrusion Detection Systems 13
1.3.1 Traditional Signature-Based Systems 13
1.3.2 Machine Learning Approaches 14
1.3.3 Deep Learning Challenges 14
1.4 The Proposed DSB-DNN Architecture 15
1.4.1 Dove Swarm Optimization 15
1.4.2 Squeeze-and-Excitation Blocks 15
1.4.3 Hybrid Architecture Design 17
1.5 Research Motivation 17
1.6 Problem Statement 18
1.7 Research Objectives 18
1.8 Significance of the Study 19
1.9 Research Contributions 20
CHAPTER 2: LITERATURE SURVEY **24**
2.1 Literature Review 24
2.2 Machine Learning Approaches for Web Attack Detection 24
2.2.1 General Web Attack Detection Frameworks 24
2.2.2 Targeted Attack and Vulnerability Detection Using Machine Learning 25
2.3 Deep Learning Approaches for Web Attack Detection 28
2.3.1 End-to-End and Unsupervised Detection 28
2.3.2 Advanced Deep Learning Architectures 29
2.4 Surveys and Broader Perspectives on Web Security 32
2.5 Identified Challenges and Gaps in the Current Literature 33
2.6 Emerging Trends and Future Research Directions 36
2.7 Challenges in Detecting Vulnerabilities 38
2.8 Limitations of Existing Detection Mode 40
2.9 Problem Definition 41
2.10 Need for Hybrid Deep Learning Approaches 42
2.11 Research Scope 44
2.12 Summary 46
CHAPTER 3: METHODOLOGY AND FRAMEWORK FOR ATTACK DETECTION **48**
3.1 Introduction and Research Design 48
3.2 Data Foundation and Preprocessing Framework 49
3.2.1 Dataset Selection and Justification 49
3.2.2 Comprehensive Preprocessing Pipeline 50
3.3 Phase 1: Dove Swarm Optimization Framework 53
3.3.1 Theoretical Foundation and Algorithm Selection 53

3.3.2 Feature Selection Implementation and Optimization 53
3.3.3 Classification Framework and Performance Analysis 55
3.3.4 Initial Standalone DSO-Based Classification Approach 56
3.4 Phase 2: Advanced Deep Learning Architecture 56
3.4.1 Architectural Rationale and Design Philosophy 56
3.4.2 Squeeze-and-Excitation Enhanced CNN Architecture 57
3.4.3 DSO-Driven Hyperparameter Optimization 58
3.5 Advanced Class Imbalance Mitigation 59
3.5.1 Synthetic Data Generation Strategy 59
3.5.2 Cost-Sensitive Learning Integration 60
3.6 Comprehensive Training Strategy and Regularization 60
3.6.1 Advanced Training Configuration 60
3.7. Evaluation Metrics and Tools 61
3.7.1 Performance Parameters 61
3.7.2 Validation Strategy 62
3.7.3 Visualization and Comparative Tools 62
3.7.4 Libraries and Toolkits 63
3.7.5 Comprehensive Testing Protocol 63
3.8 Methodological Limitations and Mitigation Strategies 64
3.8.1 Identified Limitations 64
3.8.2 Mitigation Strategies 64
3.9 Chapter Summary and Methodological Contributions 65
CHAPTER 4: RESULTS AND ANALYSIS FOR ATTACK DETECTION 66
4.1 Testing 66
4.2 Results and Discussion 69
4.2.1 Overview of Evaluation Metrics 69
4.2.2 Model wise Performance 69
4.2.3 Network for Squeeze and Excitation 71
4.2.4 VGG19 and VGG16 73
4.2.5 EfficientNetB0 73
4.2.6 DenseNet121 73
4.2.7 ResNet50 74
4.2.8 MobileNetV2 74
4.2.9 Dove Swarm Optimization 75
4.2.10 Visual Analysis of Class-Specific Patterns 76
4.3 Discussion 76
4.4 Performance Analysis 77
CHAPTER 5 METHODOLOGY FOR VULNERABILITY PREDICTION 79
5.1 Methodology Overview 79
5.2 Data Loading and Cleaning 79
2. Feature Engineering (combined text + numeric + categorical) 80
3. Model Training (Baseline → Advanced → Hybrid) 80
4. Model Evaluation (Accuracy, F1, ROC-AUC, class-wise metrics) 80
5. Final Hybrid Ensemble Model Creation 80

6. Prediction with Confidence Scores 80
5.3 Dataset Description 80
Key Attributes in the Dataset 81
Automatically Computed Insights from Code 82
5.4 Data Preprocessing 82
5.4.1 Handling Missing Values 82
5.4.2 Combined Text Construction 83
5.4.3 Numeric Feature Extraction 83
5.4.4 Categorical Features 84
5.4.5 Feature Scaling and Encoding 84
5.5 Research Design Evolution 84
5.5.1 Purpose of the Baseline Phase 85
5.5.2 Preprocessing and Feature Engineering in Phase 1 85
5.5.3 Algorithms Used in the Baseline Stage 85
5.5.4 Training and Testing Process 85
5.5.5 Observations from Baseline Results 85
5.5.6 Limitations of the Baseline Model 86
5.5.7 Baseline Architecture 86
Components from Code 86
5.5.8 Baseline Algorithm Workflow 87
Algorithm 4.1 – Baseline Training Process 87
5.6 Limitations Observed in Phase 1 88
5.7 Phase 2 – Advanced Multi-Model Evaluation (ALL_ML_MODELS.py) 88
5.7.1 Models Used 89
Summary of Model Coverage 93
5.8 Phase 3 – Hybrid Ensemble Model (Final System) 93
5.8.1 System Architecture (Descriptive Diagram) 94
5.8.2 Flowchart of the Methodology 97
CHAPTER 6 RESULTS AND ANALYSIS FOR VULNERABILITY PREDICTION 101
6.1 Results and Analysis 101
6.2 Ranking based on Accuracy 103
6.2.1 Model Comparison across All Evaluation Metrics 103
6.2.2 Model Rankings Across All Metrics 104
6.3.3 Normalized Confusion Matrices of Top Performing Models 106
6.4 Model Training Time and Prediction Time Comparison 111
6.5 Results and Discussion 112
6.6 Overall summary 115
CHAPTER 7: CONCLUSION AND FUTURE WORK 116
7.1 Conclusion Regarding Attack Classification and Prediction 116
7.1.1 Limitations 118
7.1.2 Future Work 119
7.2 Conclusion Regarding Vulnerability Classification and Prediction 119
7.2.1 Future Work 120
CHAPTER 8: PUBLICATIONS AND REFERENCES 122

INTRODUCTION

Cloud-computing has disrupted contemporary digital infrastructure, making it possible to build large-scale and distributed web applications that serve as backbone for critical services in healthcare, e-commerce and manufacturing. But this evolution has created attack-magnets and in today's landscape web apps are that thing that is thoroughly magnetized by advanced thirst. A lot of widely know vulnerabilities and security misconfigurations that can potentially pose a catastrophic threat, like SQL Injection (SQLi), Cross-Site Scripting (XSS), authentication bugs or dumb security configuration issues, always directly cause organization-level breaches as the attacks in 2017 against Equifax and Capital One in 2019 reported. Conventional IDS systems based on signature are powerless against zero-day attacks and complex threats, also suffer the bottleneck problems of feature extraction, imbalanced class and computation costs in ML/DL research.

The phrase "web vulnerability" refers to all manner of security concerns, from attacks involving injections to session management failures that ultimately can lead to data theft, unauthorized entry or disruptions in service. The modern models such as OWASP partition them into major and minor cybersecurity related neurocognitive-like disorders trying to highlight the necessity of early recognition. The present thesis aims to overcome these challenges by the proposed Dove Swarm-Based Deep Neural Network (DSB-DNN) which is a hybrid architecture, classifying frames based on fixed-size segmenting structure via DSO for feature selection and Squeeze-and-Excitation (SE) blocks for adaptive attention in CNNs.

Problem Description and Research Gap

Web applications are challenged by increasingly dangerous threats such as SQLi, XSS, DDoS and zero-day attacks that do not have a clear real-time cure. The early warning is subtle which lead to a delay on intervening even though structural signs appear on HTTP traffic. Conventional IDS waste payload data, and ML/DL models do not have agreement on predictive features and generalizable architectures. Uncertainty remains in how to manage the issues of class imbalance, high-dimensional data, dynamic cloud environment (such as false positive and computational overhead), all of which are yet to be fully addressed.

Motivation

Web vulnerabilities, particularly in the cloud scenario are very risky, there are millions of infected with scalability of exponential rate. Lack of curative approaches emphasizes the

importance of early detection for improving survival and reducing morbidity. Biomarkers of structural damage found in HTTP traffic, however, manual analysis is also far from being efficient. This thesis exploits ML/DL on the datasets, such as MSCAD, to build scalable interpretable models that architecture proposed in this thesis follows a shift from reactive security to proactive.

Problem Statement

If you look at them as symptoms of a disease, there is no cure and diagnosis is late due to subtle symptoms despite cues in HTTP. There is the problem for conventional methods that too few traffic data are used and for ML/DL approaches where there's no concensus about features and model to achieve a reliable generalisation in dynamic clouds.

Objectives

The research work aims at;

- Examine HTTP traffic and to perform the best possible pre-process to maintain payload information while eliminating irrelevant artefacts.
- Extract quantitative characteristics such as URI patterns, special characters occurrence and session measures by means of advanced morphometry.
- Tackle the black-box character of complex ML/DL models by interpretable ensembles.

Scope

The first one is to develop such an ML/DL-supporting web-based system that can interface with MSCAD and other data sources if the analysis becomes more advanced. We study texture patterns, radiomic features and feature morphometry to identify changes associated with the threat.

Research Methodology For Work Done

The work improves the accuracy of DSB-DNN hybridisation by combining DSO to optimise and SE blocks for attention. The interpretability offered by XAI techniques such as SHAP is attractive. Results displayed 97.35% accuracy versus baselines.Supporting strategies include the use of pre-processing pipelines to process heterogenous traffic, and such model homogenization techniques as bias correction and normalization.

CONCLUSION

Outcome of Research Work

Literature dictates that there is a rigorous preprocessing needed for HTTP traffic because payload details were lost or agility in dynamic data was not the focus of current methods. This gap warrants appropriate preprocessing to capture the signatures as well as filtering out noise. Structural features such as URI length and token count are important, but current tools produce inconsistent pipelines. It is crucial that the lat- ter can be systematically extracted from relevant patterns. Current ML/DL models work very well, but they are not interpretable which makes them impossible to use! As such, objectives are based on existing needs.

Objective 1:

Analyze HTTP traffic and perform optimal preprocessing to preserve payload details and remoe irrelevant artifacts.

Achievement:

Literature highlights issues in multi-source datasets like MSCAD, including noise, variability, and non-relevant fields. Classical methods struggle with pathological payloads, while DL-based approaches are robust but data-dependent. Denoising preserves anatomical info but requires optimization. Optimal preprocessing is essential for:

- Preserving subtle patterns,
- Maintaining contrast,
- Ensuring accurate extraction,
- Reducing non-relevant effects.

Thus, the objective addresses gaps in unified pipelines, preventing loss of degenerative patterns.

Objective 2:

Extract quantitative features such as URI patterns, special character counts, and session metrics using advanced morphometry.

Achievement:

Literature shows attacks characterized by alterations in patterns and volumes. Morphometry provides statistical analysis, with segmentation influencing output. Gaps include inconsistencies and loss due to improper preprocessing. Extracting features is essential for:

- Measuring atrophy patterns,
- Region-wise comparison,
- High-quality ML inputs.

This aligns with needs for standardized biomarkers, inconsistently captured in workflows.

Objective 3:

Address the black-box nature of complex ML/DL models.

Achievement:

Literature notes high accuracy in CNNs and ensembles but limitations in interpretability and clinical explainability. Studies call for XAI and attention models. Addressing this is essential to:

- Improve trust,
- Ensure transparency,
- Provide feature-based insights,
- Enhance adoption.

This objective fills gaps in interpretable prediction, even at high accuracy.

Research Contribution

Just as the DSB-DNN framework has demonstrated such promising results in this thesis, there are multiple paths for future work and improvements: building on that framework is important to improve both technical operation and intellectual delivery for web security. This critiquing is intended to overcome current limitations, achieve scalability, and gain widerness when applicable in real world cybersecurity. A prominent direction is the multimodal integration of API logs and threat indicators for better prediction. There also exists the the current DSB-DNN model that is built over features learned from HTTP traffic datasets such as MSCAD, which contains structural and payload-level indicators of attacks. To enhance prediction capability, an avenue for future work could involve fusing API logs (e.g., RESTful services or GraphQL endpoints) with behavioural threat markings, e.g., anomalous patterns of user session time stamping and abnormal user frequency access). This would result in a richer feature space which could be leveraged to detect multi-stage attacks across the network traffic and application-layer interactions. The thinking behind this is that web applications now rely on APIs, and their vulnerabilities such as API key exposure or rate limiting bypasses quite often have a different fingerprint on how they present when compared to traditional HTTP exploits. Integrating these with threat markers, inferred from the performance of user behavior analytics which can expose correlated patterns such as a SQLi attempt and unauthorized API calls at subsequent time instances that single modality analysis could overlook. We may employ fusion approaches such as early feature-and/or late decision-level integration where we would use attention mechanisms in extension of SE blocks to learn to weigh contributions of logs embeddings generated exploiting NLP tools (like BERT) on one hand and threat markers extracted out of statistical anomaly detection expressed through the attention mechanism applied over threat marks/embedding for log lines. It would be necessary to train the improved models similar to MSCAD+APICB on augmented training datasets like mixing MSCAD with API-specific benchmarks for example (OWASP API Security Project). Potential impact would be being able to increase UV detection in hybrid threat environment by 10-15%, based on preliminary multimodal findings in the cybersecurity space, and that there would be numerous cost-savings associated with longitudinal monitoring in dynamic cloud environments, being able to predict escalation of attack chains proactively thus reducing response time in security operations center (SOC)s.

Second, generalizability across demographics and protocols would be achieved using transfer learning. The DSB-DNN works well for the MSCAD case, but its applicability not only for

different user demographics (e.g., difference in geographical origin and device type), but also difference protocols such as HTTP/3 or communication based on WebSocket is still limited. Transfer learning can help to fine-tune pre-trained models on new domains without the need for a complete re-training and improve robustness across heterogeneous web landscapes. The intuition is that threat landscape fluctuates depending on region e.g., higher phishing in some demographics of the users and protocol i.e., encrypted traffic in modern standards hides payload, that makes current model tend to overfit the data and generalize poorly when deployed through devices across environment type (such as from enterprises to IoT web apps). Potential approaches are to adapt the DSB-DNN on the domain using a technique like Domain-Adversarial Neural Networks (DANN) or unsupervised domain adaptation, preshaping on large-scale datasets such as UNSW-NB15 or CIC-IDS2017, and sharing across domains with improved multitask/ knowledge transfer layers but with considered demographics-aware augmentations capturing regional traffic behaviours by synthetic data generators similar to GANs. Among expected benefits are 5-10% accuracy improvements in out-of-set protocols (as reported for other transfer learning applications in network security), better deployability of the framework in global deployments, adaptiveness of the security to different user populations, and a lower need to retrain from scratch.

To handle the compute-intensive optimization, parallel GPU-based implementations are recommended. The DSO element in the DSB-DNN produces high computational burden in the hyperparameter tuning, and feature selection especially when utilizing large datasets to be trained upon, a GPU parallelized implementation may remove this obstacle by fostering convergence speedup and scalability guaranteeing real-time applications. The motivation is that computing optimization stages such as swarm iterations over high-dimensional spaces can be costly with slower deployment on resource- or time-constrained platforms and GPU parallelization methods to distribute the computation of swarming to allow decreasing in training times without degrading the optimization effectiveness. Possible approaches are the reimplementation of DSO with CUDA compatible libraries such as CuPy or PyTorch for parallel particle updates and fitness evalutions, batched-parallel swarm operations where multiple doves (particles) are operated on in parallel across GPU cores, benchmarking against CPU baselines on larger datasets, load balancing for heterogeneous hardware. Intended outcomes would include up to 50%-70% reduction in optimization time, according to GPU acceleration benchmarks observed in metaheuristic algorithms; enabling production on-the-

fly retraining and rendering DSB-DNN practical for edge computing in web security where fast response to new threats is a must.

Federated learning presents an opportunity for distributed models to be developed in a decentralized manner without the need to transmit data. Centralized training using sensitive security data has serious privacy issues with proprietary logs in datasets like MSCAD, whereas federated learning (FL) can facilitate collaborative model training across distributed nodes (e.g., multiple cloud instances) without sharing the original data, thus keeping confidentiality while enhancing model performance. The reason is that web security data is frequently stored in siloed from different organizations or geographies due to regulations like GDPR and FL supports aggregating model updates instead of raw data, which reduces the risk while benefiting from diverse threat intelligence sources. Potential Future Work Our work will be extended extending the DSB- DNN to FL frameworks such as TensorFlow Federated or PySyft, where local DSO optimizations are allowed at edge nodes and global SE-CNN parameters are aggregated centrally; integrating differential privacy for adding noise in updates and also avoiding inference attacks, with testing on simulated federated setups using partitioned MSCAD subsets. Anticipated benefits include greater model diversity with diverse attacks vectors from distributed data; possible improvements in accuracy of 8-12% in the multi-domain setting; and the ability to perform privacy-preserving collaborations (e.g., across enterprises for a joint defense against new strains of web threats).

Integrating into clinical workflow for real-time support and operations would integrate research to practice. Although DSB-DNN performs well at the offline assessment stage, its operationalization has not been investigated, and future work could aim to incorporate the framework into live security applications (e.g., SIEM systems or WAFs) for alert feedback and action automation. The logic is that the adoption of research models very often strives while stuck in experimental forms and have gaps in deployment, ease of integration into operations would fill these gaps, making possible effective threat management in real environments and verifying if the framework can be useful in practice. Possible approaches may also involve creating plugins for analytics tools (e.g. Splunk, ELK stack), with DSB-DNN processing live traffic flows and/or deployment of real-time inference optimizations such as model quantization and closed feedback loops to enable continuous learning, with pilot deployments taking place in emulated enterprise environments which monitor detection latency and false positive rates among other measures. Anticipated Impact The resulting benefits would be to reduce mean time to response (MTTR) by 30-40%, elevating DSB-DNN from a tool for

research to an operational capability; enabling user studies with security analysts, so as to improve interpretability attributes such as SHAP visualizations for more effective decision support.

Objective 4 Vulnerability Prediction using Hybrid ensemble model and comparison of multi model approaches.

The workflow begins with raw web attack logs, which represent real HTTP request data collected from web traffic. These logs include request methods, URIs, user-agent strings, payload data, severity levels, and geographic information. Since raw data often contains inconsistencies and missing values, it cannot be directly used for machine-learning tasks.

In the next stage, data preprocessing is performed. This step focuses on cleaning the dataset by handling missing values, standardizing text fields, and ensuring consistency across all attributes. Proper preprocessing ensures that noisy or incomplete records do not negatively affect model learning and evaluation.

After preprocessing, the system performs feature engineering, where meaningful features are extracted from the cleaned data. This includes:

Textual features derived from request components using TF-IDF,

Numeric features such as length measures and special character counts,

Categorical features like HTTP methods and country codes encoded using one-hot encoding.

All these features are combined into a unified feature matrix, allowing the models to capture both semantic and structural attack patterns.

The processed features are then passed to Phase 1: Baseline Model Training. In this phase, simple machine-learning models are trained to establish an initial performance benchmark. This step helps in understanding dataset behavior and validating the effectiveness of feature engineering.

Next, the framework moves to Phase 2: Multi-Model Evaluation, which represents the core experimental phase of the research. Here, twelve different machine-learning algorithms are trained and evaluated using a common preprocessing pipeline. Multiple evaluation metrics and cross-validation techniques are applied to compare model performance in terms of accuracy, stability, and class-wise behavior.

Based on the results obtained in Phase 2, the system advances to Phase 3: Hybrid Ensemble Model Construction. In this phase, the top three performing models are selected and combined using an ensemble strategy. This hybrid approach leverages the strengths of individual models while reducing their weaknesses, resulting in improved prediction reliability and robustness.

Finally, the trained hybrid ensemble produces the final attack prediction along with confidence scores. These confidence values indicate the certainty of each prediction, enabling better decision-making for security analysts and allowing prioritization of high-risk attacks.

Overall, this architecture demonstrates a systematic, code-driven, and research-oriented approach to web-attack detection, progressing logically from raw data to accurate and confident attack classification.A web application was designed so that when we provide various HTTPS Field data,it predicts the vulnerabilities present and the best model that predicts it.

FUTURE WORK

Based on the encouraging results obtained in this thesis, following up with multiple possible directions for future research and development will continue to improve DSB-DNN for securing web applications. Such extensions are to alleviate the existing limitations, to enhance scalability and further bring towards real-world practicalcyber security scenarios. An important direction is the fusion with API logs and threat indicators for better prediction. Current DSB-DNN model is depend on analyzing HTTP traffic based datasets such as MSCAD which has structural and payload-based characteristics of attacks. In order to increase predictive power, multimodal data fusion could be leveraged in future work which combines API logs (e.g., of RESTful services and/or of GraphQL endpoints) with behavioral threat markers like suspicious user session patterns or irregular access frequencies. This provide a broader basis of the feature space and facilitates the identification of multi-step attacks through network traffic and application-level interactions. This technique is based on the premise that web applications are relying more and more on APIs, with things like API key exposure or rate limiting bypasses presenting in different ways than traditional HTTP vulnerabilities. Fusing these together with threat markers obtained from user behavior analytics can potentially expose correlated patterns such as a SQLi attempt followed by unauthorized API calls that single modality based analysis could overlook: possible avenues would be to use fusion techniques such early feature-level or late decision-level integration - e.g., adding attention mechanism in widening 3,4 SE blocks for weighing the contributions of embedding of API logs generated using NLP tools like BERT and those from threat marker extracted through statistical anomaly detection. It would be necessary to train the enhanced model on augmented datasets, like merging MSCAD with API-specific benchmarks including data from the OWASP API Security Project. Anticipated impacts are the enhancement of detection accuracy by 10-15% for hybrid threats, as derived from prior multimodal work in cybersecurity, and to enable longitudinal monitoring in a highly dynamic cloud environment to proactively forecast burgeoning attack chains that help speed up response times in SOCs.

" It is also an important for future work to address transfer learning across demographics and protocols. DSB-DNN shows good performance on MSCAD, but it cannot generalize to users with various geographies and devices and new unlimited protocols like HTTP/3 or WSS-based communication. Transfer learning may allow to adapt pre-trained models to new domains with little or no retraining, and to be robust across diverse web ecosystems. The reason for this is that the space of cyber-threats are different across regions (e.g., higher phishing rates among

certain demographics) and protocol-level (current encryption on modern standards make payloads obscure in encrypted traffic); models today may overfit to specific datasets and perform poorly on deployments across environments, say from enterprise to IoT web apps. Possibility methods can be to fine-tune the DSB-DNN using domain-adaptive approaches (e.g., Domain-Adversarial Neural Networks, UNSW-NB15) or unsupervised domain adaptation (CIC-IDS2017), pre-train on large-scale datasets, including UNSW-NB15 and CIC-IDS2017, and transfer itself acquired knowledge to target domains through exploitation of feature alignment layers whilst leveraging demographic-aware augmentations (e.g., simulating regional traffic dynamics by using synthetic data generators like GANs). Possible benefits are that the model would likely achieve 5-10% increase in accuracy on novel protocols, as observed with other transfer learning investigations in network security. Making the framework more usable at global scale, providing adaptive security for all classes of users and eliminating the need for custom retraining should also be expected benefits.

Parallel GPU implementations are recommended to reduce the computational burden in optimization. The DSO in the proposed DSB-DNN also poses high computation overhead, namely hyperparameter tuning, and feature selection (tabulated in Table 2), especially for large datasets; scalable parallelizable GPU-based implementation could be employed to minimize this complexity ensuring faster convergence and scalability for real-time applications. The reason is that optimization phases with swarm iterations through high-dimensional spaces are computationally demanding, and deployment in resource-constrained environments may be restricted; whilst the CPU-GPU Parallelism can share the swarm computation in-between, leading to training time reduction but without affecting on the quality of the optimization results. Possible approaches include refactoring DSO with CUDA-backed libraries such as CuPy or PyTorch to enable parallel updates and fitness evaluations of particles, utilizing batched-parallel swarm operations where multiple doves (i.e. progress particles) are updated simultaneously on GPU cores, comparing with CPU baselines using a wider range of datasets while load-balancing for heterogeneous hardware. The anticipated gain may cut down on optimization time by ahead-of-time retraining for 50 to 70%, according to our GPU acceleration benchmarks of metaheuristic algorithms, opening the way to DSB-DNN deployment as an edge-computing service in web security, where quick adaptation face new threats are crucial.

Federated learning enables decentralized models with distributed data without exchange. Trained centrally on sensitive security data also raises privacy issues, as the datasets such as

MSCAD may consist of proprietary logs and federated learning (FL) would allow collaborative model training across distributed nodes (e.g., cloud instances), without direct sharing of raw data but improved confidentiality while providing higher robustness on model. The argument is that since web security data can be siloed organizationally or regionally for compliance reasons (like GDPR), and FL allows sharing model updates rather than raw data, it reduces risk while pooling in threat intelligence from multiple sources. Possible approaches include porting DSB-DNN to FL settings such as TensorFlow Federated or PySyft, where local DSO optimizations are done on edge nodes while global SE-CNN parameters are averaged at the centre, integrating differential privacy to add noise to updates and protect against inference attacks, and experimenting on synthetic federated set-ups that simulate partitioned MSCAD subsets. The potential benefits are improved model diversity that integrates different attack vectors scattered in data and which could yield an accuracy improvement of SK-12% in the multi-domain case, as well as facilitating privacy-preserving sharing among colluding enterprises for collective defense of dynamic web threats.

Ultimately, integration of PTHP into workflows for real time support would close the gap between research and implementation. Although DSB-DNN performs well for offline evaluation, its operationalization in real world scenarios has not been investigated and developing methods to incorporate the framework into live security pipelines (eg., SIEM systems or WAFs) for actionable alert notification generation and automatic reactions would be future work. The justification is that many models stop at the experimental phase due to the gap between its deployment and operationalization, with operational integration filling it by allowing a real-time suppress of threats to happen and testing this proof-of-concept scenario in practice. Pipeline strategies include writing plugins for tools such as Splunk or ELK stack in which DSB-DNN is applied over incoming traffic streams might involve integration with current and future real-time inference optimizations (e.g., model quantization, feedback loops for continuous learning); and pilot deployments of components under SIM pr to report detection latency and false positive rates in simulated enterprise environments. The anticipated payoffs will lower covariance and mean time to response (MTTR) by 30-40%, turning DSB-DNN tool from a research prototype into an operational asset, enabling user studies with security analysts, improving interpretability such as SHAP visualization for further decision support.

BIBLIOGRAPHY

1. M. Betarte, J. P. Campo, F. Kirchner, and V. Russi, "Directly connecting one-class and multi-class machine learning models to ModSecurity for SQLi and XSS detection," Journal of Computer Security, vol. 26, no. 1, pp. 21–42, 2018.

2. S. Mighan and M. Kahani, "A comprehensive review of machine learning and deep learning for intrusion detection systems and feature selection in web vulnerabilities," Cybersecurity Review, vol. 7, no. 1, pp. 112–134, 2024.

3. R. Gupta, S. Kumar, and A. Singh, "Machine learning for attack and flaw detection in web server logs," in Proc. Int. Conf. Cyber Security (ICCS), 2022, pp. 45–53.

4. T. Nguyen, M. Nguyen, and T. H. Nguyen, "Ensemble techniques for detecting XSS flaws in online applications," IEEE Access, vol. 9, pp. 12054–12067, 2021.

5. M. Hasan, M. Nasser, B. Sen, and M. S. H. Chowdhury, "CNN and LSTM models for SQL injection detection in web forms," IEEE Transactions on Reliability, vol. 71, no. 2, pp. 881–894, 2022.

6. Y. Lee and J. Kim, "Semi-supervised machine learning for DDoS attack detection in web servers," Computers & Security, vol. 77, pp. 212–225, 2018.

7. S. Gupta, A. Bakshi, and N. Golecha, "Applying ML/DL methods to find phishing attacks via URL weak points," International Journal of Information Security, vol. 19, pp. 433–450, 2020.

8. M. Aljabri et al., "Analyzing ML and DL algorithms for phishing discovery in web interfaces," IEEE Access, vol. 11, pp. 15432–15448, 2023.

9. X. Sun, J. Wang, and Y. Zhang, "Adversarial ML for detecting spam and phishing on social media," Neural Computing and Applications, vol. 33, no. 14, pp. 8121–8136, 2021.

10. A. Sharma, S. Kumar, and P. Gupta, "Machine learning for session-based vulnerability detection," in Proc. IEEE 6th Int. Conf. Softw. Secur. Rel., 2012, pp. 102–110.

11. K. Patel and H. Shah, "Machine learning methods for authentication vulnerability detection in web applications," Security and Privacy, vol. 6, no. 3, p. e284, 2023.

12. S. Bhadauria and R. Prasad, "Predicting web attacks and session hijacking using machine learning," Journal of Information Security and Applications, vol. 42, pp. 12–24, 2018.

13. R. Singh and J. Kaur, "Support Vector Machines and Decision Trees for CSRF detection," Procedia Computer Science, vol. 201, pp. 556–563, 2022.

14. S. Hou et al., "Deep learning and system call graphs for Android web-based malware detection," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 12, pp. 3314–3327, 2016.

15. J. Wang, R. Zhao, and K. Chen, "Combining static analysis and machine learning for web vulnerability detection," ACM Transactions on Privacy and Security, vol. 21, no. 2, pp. 1–28, 2018.

16. B. Park, Y. Kim, and S. Kim, "ML models for executable analysis in web exploit detection," Journal of Web Engineering, vol. 18, no. 4, pp. 305–326, 2019.

17. Q. Tran and T. Le, "Deep learning techniques for web danger information extraction," Information Sciences, vol. 621, pp. 450–468, 2023.

18. H. Kim, S. Kim, and T. Kim, "Byte-based Deep Neural Networks for web application malware classification," IEEE Transactions on Information Forensics and Security, vol. 14, no. 11, pp. 2854–2867, 2019.

19. P. Singh and M. Kumar, "Deep learning for zero-day web malware detection," Computer Networks, vol. 238, p. 110098, 2024.

20. S. Ahmed, J. Lee, and K. Park, "A detailed assessment of DL for malware detection on desktop and mobile devices," IEEE Communications Surveys & Tutorials, vol. 26, no. 1, pp. 512–545, 2024.

21. Z. Tian et al., "Unsupervised detection of attack traces using RSMT and autoencoders," IEEE Transactions on Industrial Informatics, vol. 15, no. 12, pp. 6512–6521, 2019.

22. L. Wang, Y. Zhang, and X. Liu, "Autoencoder-based algorithms for online traffic vulnerability detection," Frontiers in Computer Science, vol. 3, p. 642011, 2021.

23. I. Sarker et al., "Hybrid CNN and Random Forest models for web attack detection," Applied Intelligence, vol. 54, no. 2, pp. 1845–1862, 2024.

24. Y. Chen, S. Zhao, and L. Huang, "Hybrid LSTM and CNN models for session stealing detection," Knowledge-Based Systems, vol. 265, p. 110344, 2023.

25. J. Kim, H. Choi, and S. Park, "Convolutional Channel-BiLSTM Attention (CCBA) for command injection detection," IEEE Access, vol. 12, pp. 5432–5445, 2024.

26. X. Zhao, Y. Feng, and J. Wu, "Graph Neural Networks for modeling web application vulnerability connections," Expert Systems with Applications, vol. 190, p. 116210, 2022.

27. D. Patel, M. Mehta, and R. Joshi, "CNNs for detecting misconfigured security headers," International Journal of Computer Networks and Applications, vol. 11, no. 1, pp. 1–15, 2024.

28. V. Kumar, K. Sharma, and R. Singh, "LSTM-based models for real-time web attack discovery," Real-Time Systems, vol. 60, no. 2, pp. 245–268, 2024.

29. F. Patel, K. Patel, and G. Patel, "DL-based packet analysis for Web Application Firewalls," IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 1120–1135, 2023.

30. M. Ahmed, S. Ali, and N. Khan, "ML-enhanced WAFs for SQLi and XSS security gaps," Journal of Information Security, vol. 14, no. 4, pp. 312–330, 2023.

31. S. Ali and H. Zhang, "DL approaches for automated security flaw discovery," Software Engineering Notes, vol. 48, no. 1, pp. 22–34, 2023.

32. R. Ali, M. Khan, and S. Iqbal, "Automating penetration testing using deep learning," Automation in Software Engineering, vol. 31, no. 2, pp. 89–105, 2024.

33. H. Chen, S. Liu, and W. Wang, "Machine learning for threat discovery in multimedia networks," Multimedia Tools and Applications, vol. 77, no. 15, pp. 19543–19561, 2018.

34. S. Kumar and V. Gupta, "Random Forest and XGBoost for uncovering unsafe deserialization," Security and Communication Networks, vol. 2023, p. 8854321, 2023.

35. A. Alsaedi et al., "Semi-supervised learning methods for intrusion detection systems," IEEE Access, vol. 11, pp. 24567–24580, 2023.

36. Y. Zhang, M. He, and G. Shi, "Deep Belief Networks for network-based web threat detection," Neural Processing Letters, vol. 51, pp. 1543–1558, 2020.

37. Z. Liu et al., "A review of 63 deep learning projects for web vulnerability detection," Journal of Systems and Software, vol. 175, p. 110915, 2021.

38. M. Alsheikh, S. Lin, and D. Niyato, "Deep learning methods for DDoS detection in IoT environments," IEEE Internet of Things Journal, vol. 12, no. 2, pp. 1234–1255, 2025.

39. F. Rahman, R. Islam, and T. Ali, "Internet measurement methods for online attack discovery," Network Security, vol. 2023, no. 8, pp. 6–14, 2023.

40. K. Kumar, S. Rathore, and J. Park, "Deep learning classifiers for XSS and SQLi vulnerability detection," International Journal of Machine Learning and Cybernetics, vol. 14, pp. 1221–1238, 2023.

41. L. Chen, Y. Wang, and J. Zhang, "Transfer learning for cross-domain web problem detection," Applied Soft Computing, vol. 145, p. 110567, 2023.

42. V. Gupta, S. Singh, and M. G. Khan, "Explainable AI (XAI) for machine learning trust in cybersecurity," IEEE Transactions on Artificial Intelligence, vol. 6, no. 1, pp. 45–60, 2025.