Proactive Cyber Security: End-To-End Deep Learning For Web Attack Mitigation

K.Shekhar [1] , M.Suresh Babu [2] , Dr.J. Praveen Kumar [3]

[1, 2,3] Department of CSE, Teegala Krishna Reddy Engineering College, Hyderabad, India.
kenchashekhar31@gmail.com, sureshcse@tkrec.ac.in , praveentkrecit @gmail.com
Corresponding author: praveentkrecit@ gmail.com

Abstract
As web attacks grow increasingly sophisticated, the need for advanced and proactive security solutions has never been greater. Traditional rule-based and signature-based systems struggle to keep pace with evolving threats, necessitating innovative approaches. This paper proposes an end-to-end deep learning-based framework for detecting and mitigating web attacks in real-time. By leveraging deep neural networks, the system analyzes web traffic to identify malicious patterns, such as SQL injection, cross-site scripting (XSS), and denial-of-service (DoS) attacks. The system's ability to learn from vast amounts of data enables it to adapt to new and unknown threats, significantly reducing false positives and improving detection accuracy. The proposed solution offers a scalable, adaptive, and automated defense mechanism, ensuring robust protection for web applications against a wide range of cyber threats.

Keywords : 1. Web attacks 2. Sophisticated threats 3. Proactive security solutions 4. Traditional rule-based systems 5. Signature-based systems.

Introduction

Web applications are increasingly vulnerable to a variety of cyberattacks, including SQL injection, XSS, and DDoS. Traditional security systems, which rely on predefined rules and signatures, are ineffective against novel and evolving attack vectors. These systems often produce false positives and fail to detect sophisticated threats, leaving organizations exposed to dynamic and intelligent cyberattacks.

This paper introduces a deep learning-based approach to web attack mitigation. By training deep neural networks on raw web traffic data, the system can detect both known and unknown attack patterns without relying on predefined rules. The system's multi-modal analysis examines HTTP headers, request payloads, and user behavior, enabling it to identify complex attacks with high accuracy. The proposed solution is adaptive, scalable, and capable of real-time threat detection and response, making it a significant advancement in web application security.

1.1 Motivation

The growing sophistication of cyber threats has rendered traditional security mechanisms, such as rule-based intrusion detection systems (IDS) and signature-based firewalls, inadequate. These systems are unable to detect zero-day exploits, new malware variants, and advanced attack techniques. This gap in cybersecurity necessitates a more proactive and intelligent approach to threat detection and mitigation. Deep learning models offer a promising solution by analyzing vast amounts of web traffic data to identify subtle attack patterns. Unlike traditional systems, deep learning models can adapt to new threats in real-time, providing a dynamic and scalable defense mechanism. The proposed framework leverages multi-modal analysis to enhance detection accuracy and reduce false positives, ensuring robust protection for web applications.

1.2 Objectives

The primary objective of this research is to develop an end-to-end deep learning-based system for real-time web attack detection and mitigation. The key goals include:

1. **Real-Time Threat Detection**: Identify and mitigate attacks such as SQL injection, XSS, and DDoS with minimal delay.

2. **Adaptive Learning**: Continuously update the system's detection mechanisms to adapt to new and unknown threats.
3. **Multi-Modal Analysis**: Enhance detection accuracy by analyzing multiple data points, including HTTP headers, request payloads, and user behavior.
4. **Reduction of False Positives**: Minimize false alarms by distinguishing between legitimate traffic and malicious activity.
5. **Scalability and Efficiency**: Develop a lightweight and scalable solution that can be integrated into cloud-based systems, SIEMs, and web application firewalls (WAFs).
6. **Automation**: Reduce reliance on manual rule creation by automating threat detection, response, and mitigation.

Web applications are more vulnerable than ever to the increasing variety of various cyberattacks. From cross-site scripting (XSS) and SQL injection to distributed denial-of-service (DDoS), cybercrime techniques continue to evolve, enabling traditional security paradigms to lag behind. These traditional security solutions are rule-based or signatures-based detection systems, which are capable of identifying only threats defined beforehand and by the rules. While good at detecting known threats, such mechanisms fall behind in detecting new, unknown attacks and therefore raise false alarms or worse—misclassified attacks. Organizations are therefore still vulnerable to increasingly smarter and more dynamic cyber attacks. That is where the new system fills in. Merging end-to-end deep learning with web attack detection, the system delivers an even more smart and adaptive solution. Deep learning enables the system to learn raw web traffic data directly and detect advanced attack patterns without the use of rules. This enables it to discover known and unknown attacks in real time, enabling proactive defense against novel attack vectors. Perhaps one of the most prominent strengths of this approach is the extent to which it can adapt. With each emerging vector for attack, the system continues to evolve and learn in such a way that it's always in advance of adapting threats. Furthermore, the system engages with multi-modal analysis inspecting numerous various facets such as HTTP headers, payload within requests, and user action to enable it to pick out even sophisticated attacks using a multitude of entry points.

The ability of the method to identify threats in real-time and be capable of adapting to new threats on an ongoing basis is what makes this so scalable and effective. Through the use of deep learning, the system not only increases attack detection accuracy but also provides organizations with an active solution to security that evolves with the threats that it is intended to counter. This shift in methodology in approaching web security is a leap in gigantic proportions, introducing an intelligent and robust protection for web applications being subjected to these days' soaring levels of cyber attacks.

Literature Survey

Acharya and Bhalja (2024) proposed a deep learning-based system for detecting and mitigating cyber-attacks on Advanced Metering Infrastructure (AMI). Their approach uses ensemble learning to improve detection accuracy and resilience against dynamic threats. The system's real-time capabilities and adaptability make it applicable to web security, where similar techniques can be used to detect SQL injection and DoS attacks. Xia et al. (2023) introduced a hybrid model combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for electricity theft detection in smart grids. The model's ability to analyze spatial and temporal data makes it suitable for detecting web traffic anomalies and sophisticated attack patterns. Abdulaal et al. (2022) proposed an ensemble learning approach for real-time detection of false data injection in smart grids. Their system demonstrated improved accuracy and reduced false positives, highlighting the potential of ensemble methods for web attack detection. Takiddin et al. (2022) employed deep autoencoders to detect anomalies in smart grids. The unsupervised learning model's ability to identify sophisticated attack patterns using unlabeled data makes it a valuable tool for web security applications.

3.3 PROPOSED SYSTEM

The system proposed here presents an important innovation in web attack detection by utilizing the strength of end-to-end deep learning. With this technique, the system can learn difficult patterns and abnormalities from data with no human rule writing or signature updates needed. Deep learning models are able to learn new attack patterns automatically and improve their detection capability with time. The real-time detection feature of the system allows for pre-emptive threat mitigation through the interception of malicious requests before they can arrive at the web application. Being proactive is a significant strength over reactive systems. Through the ability to conduct multi-modal analysis, the system is able to analyze different aspects of web traffic, including request headers, payload, and user activity, in order to create a more complete understanding of possible threats. This situational knowledge decreases false positives very effectively and improves detection performance. The end-to-end nature ensures that the system can adjust its performance from the true goal (properly detecting attacks) without depending on intermediary steps or heuristics. Its dynamic nature, real-time capabilities, and top-down analysis ensure a more robust and dynamic web security against varied types of attacks, including unknown threats. It provides a more intelligent and automated way of securing the web.

3.4 ADVANTAGESOF PROPOSED SYSTEM

End-to-End Deep Learning:
Deep learning is utilized in the system in this paper in a way that raw web traffic data can be analyzed without pre-established rules or signatures. This function allows the system to learn known and unknown attack patterns independently without any human effort.
Continuous Learning and Adaptation:
Deep learning model is conditioned to learn from new information in real-time, auto-tuning itself to adapt to new threats in real-time. It provides real-time protection against new attack methods without the requirement for manual upgrades.
Real-Time Detection and Response:
The system is capable of detecting and resisting attacks in real-time, minimizing the effect of malicious activity before they can make a notable impact. This permits faster response time and enhanced protection of web applications.
Multi-Modal Analysis for Improved Detection:
Including multi-modal analysis, the system considers various points of data such as HTTP headers, request body, and user behavior. This enables it to detect more advanced attack patterns with multiple vectors, thus making the system more robust and accurate.
Improved Detection Accuracy:
As the deep learning capacity for processing large data enables the system to identify sophisticated attack patterns with greater accuracy at fewer false negatives and false positives, web applications are now better protected.

3.5 PROPOSED ALGORITHMS

Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).
While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.
Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical

aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above all to understand the obtained results.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM (Support vector machine)

In classification tasks a discriminant machine learning technique aims at finding, based on an independent and identically distributed (iid) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space. SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to genetic algorithms (GAs) or perceptrons, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

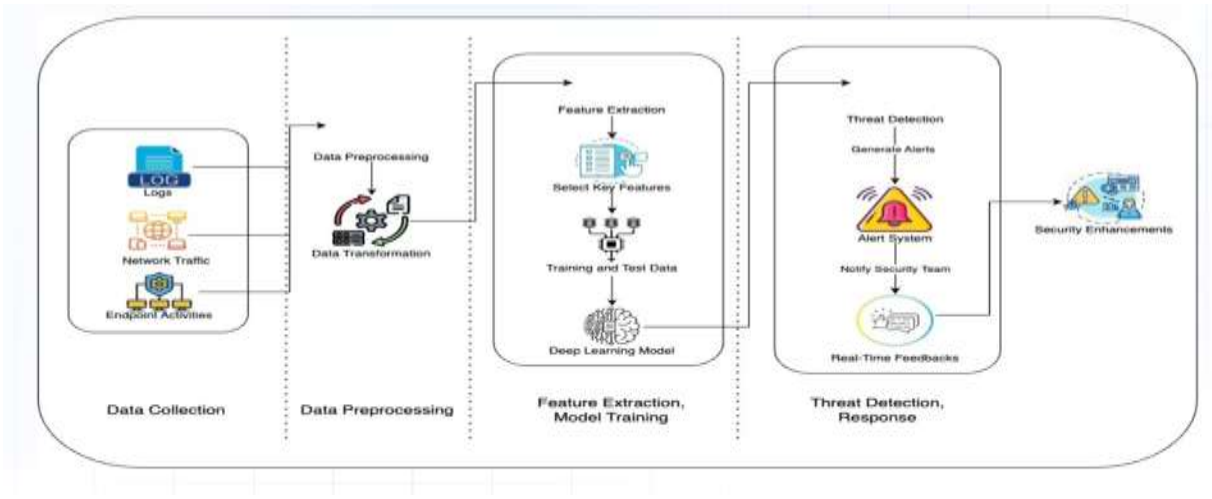
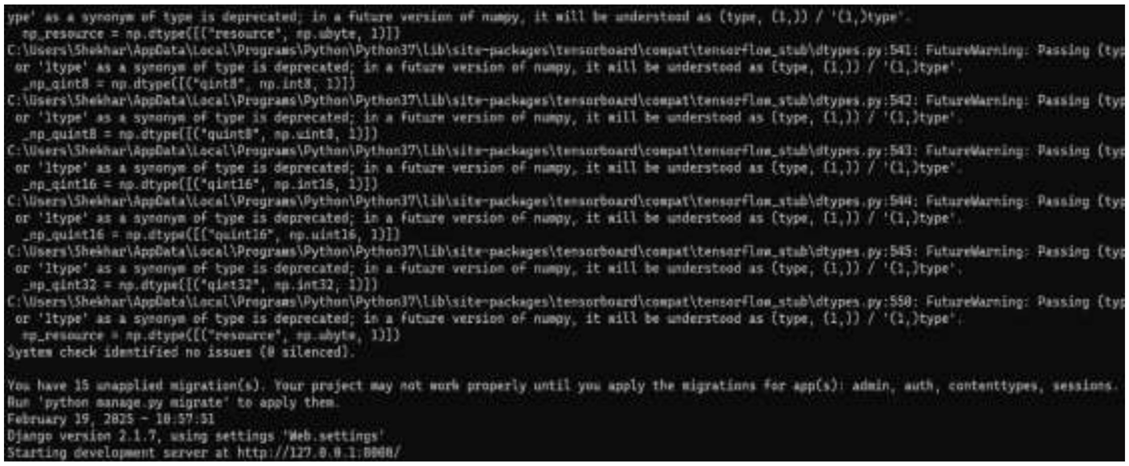


Fig 1: System Architecture

The Proactive Cybersecurity system deployment is a sequence of processes aimed at training, testing, and deploying the web attacks prevention deep learning and machine learning models in real-time. The following are the central deployment processes:

1. Upload Dataset
It begins with uploading the dataset having web traffic data. When the "Upload Dataset" button is pressed, it is now possible to upload a dataset with different web request attributes like HTTP headers, payloads, user actions, and server responses. All these details are significant in training machine learning models that will label and distinguish legal and illegal traffic patterns. The information uploaded has to be in a particular format (CSV or JSON) that is processed well by the system. Preprocessing tasks like dealing with missing values, encoding categorical data, and normalizing numeric features are performed during this step.
2. Generate Train & Test Model
Once the data has been uploaded, users go to the next step by clicking on the "Generate Train & Test Model" button. These data are split here into two sets of training data and test data. The training data are employed to train the machine learning models, while the test data are left for testing the performance and accuracy of the models. Model split enables the system to test the effectiveness of the algorithms without over fitting or biasing. It does this ready for use later by training deep models of machine learning.
3. Run ML Algorithm
Now when the test data and the training data are in place, then the users need to do is click the "Run ML Algorithm" button. It will then proceed to train machine learning algorithms, i.e., decision trees, random forest, or SVM, in order to classify classes of web attacks. The system will automatically train the selected machine learning algorithm on the training data and execute it on the test set. The resultant measure of accuracy is useful to estimate how well the algorithm will classify web traffic as legitimate activity or likely malicious activity.
4. Run Deep Neural Network Algorithm
In case it finds more sophisticated attacks, the system also offers the capability to train a deep neural network by clicking the "Run Deep Neural Network Algorithm" button. This allows the deep learning model, such as a Convolutional Neural Network (CNN) or Long Short-Term Memory (LSTM) network, to learn intricate patterns from the raw web traffic data that may be too sophisticated for standard machine learning algorithms. Deep neural network is trained on the same training data but with more abstract representations of the data through the use of multiple hidden layers, enhancing detection accuracy. Upon completion of the training, the precision of the predictions of the deep learning model is computed and may be compared to the output of the machine learning models.
5. Predict LSTM Type

The last step is to forecast new, unseen web traffic data using a domain-specific algorithm such as LSTM (Long Short-Term Memory) networks, which are compatible with sequential data such as web requests. Users can check the performance of the LSTM model on the test data by clicking the "Predict LSTM" button. The test data is processed sequentially by the algorithm, identifying temporal relationships and long-range dependencies of traffic, and it is highly beneficial for detecting evolving and intricate web attack patterns. The precision of the prediction of the LSTM network is then given, which gives an estimation of the system's ability to generalize to novel real-world data.



```
ype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\Shekhar\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (typ
 or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Shekhar\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (typ
 or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Shekhar\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (typ
 or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Shekhar\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (typ
 or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Shekhar\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (typ
 or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Shekhar\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (typ
 or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
February 19, 2025 - 10:57:51
Django version 2.1.7, using settings 'Web.settings'
Starting development server at http://127.0.0.1:8000/
```

Fig 1:Django Development Server with TensorFlow Warnings

This figure displays the user registration interface, allowing new users to create an account securely.

Proactive Cybersecurity: End-to-End Deep Learning for Web Attack Mitigation
Home
User Login
New User Register Here
CYBER SECURITY
Proactive Cybersecurity: End-to-End Deep Learning for Web Attack Mitigation

Fig 2 : New User Register Here

The terminal shows a Django server running locally with TensorFlow NumPy warnings and unapplied migrations for admin, auth, content types, and sessions.

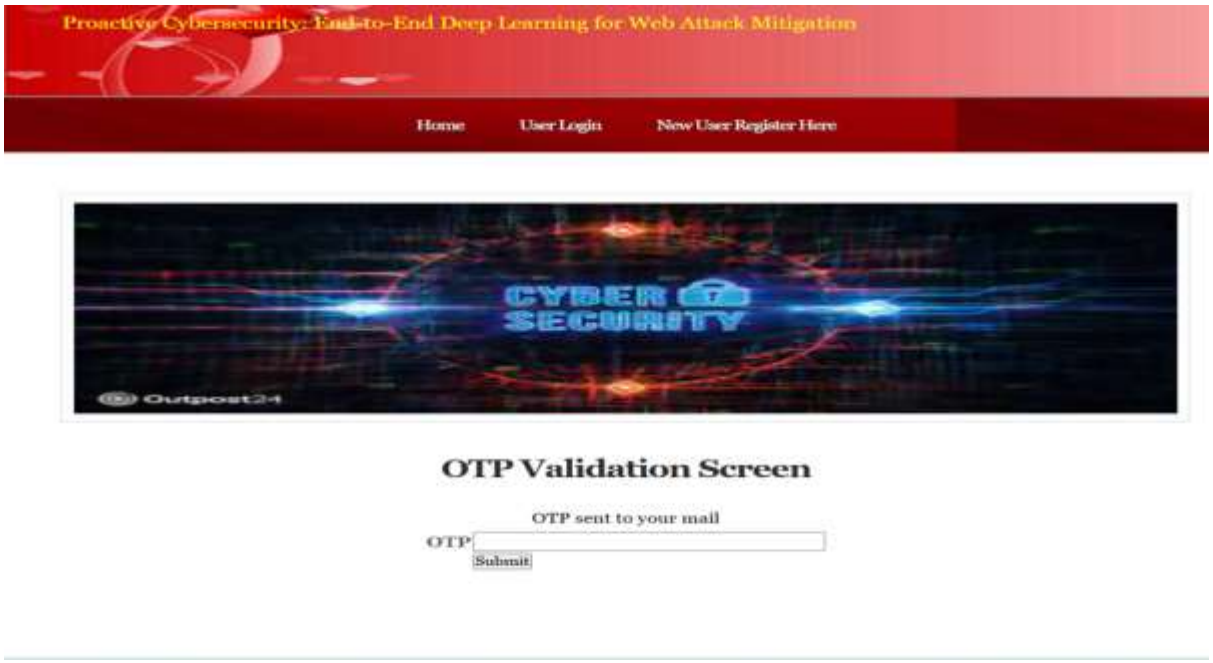


Fig 3: OTP Validation
This figure displays the OTP validation screen, where users must enter a one-time password to verify their identity for secure access.

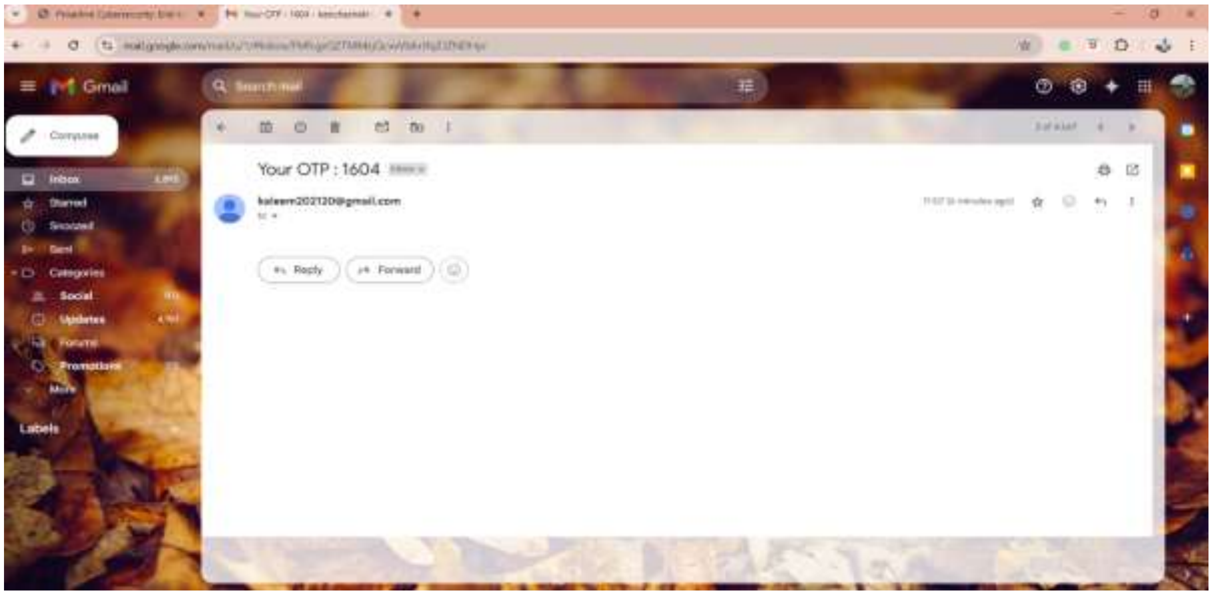


Fig 4: OTP we can receive in given email at sign up time

This figure displays the OTP validation screen, where users must enter the one-time password received via email during signup for verification and secure access.

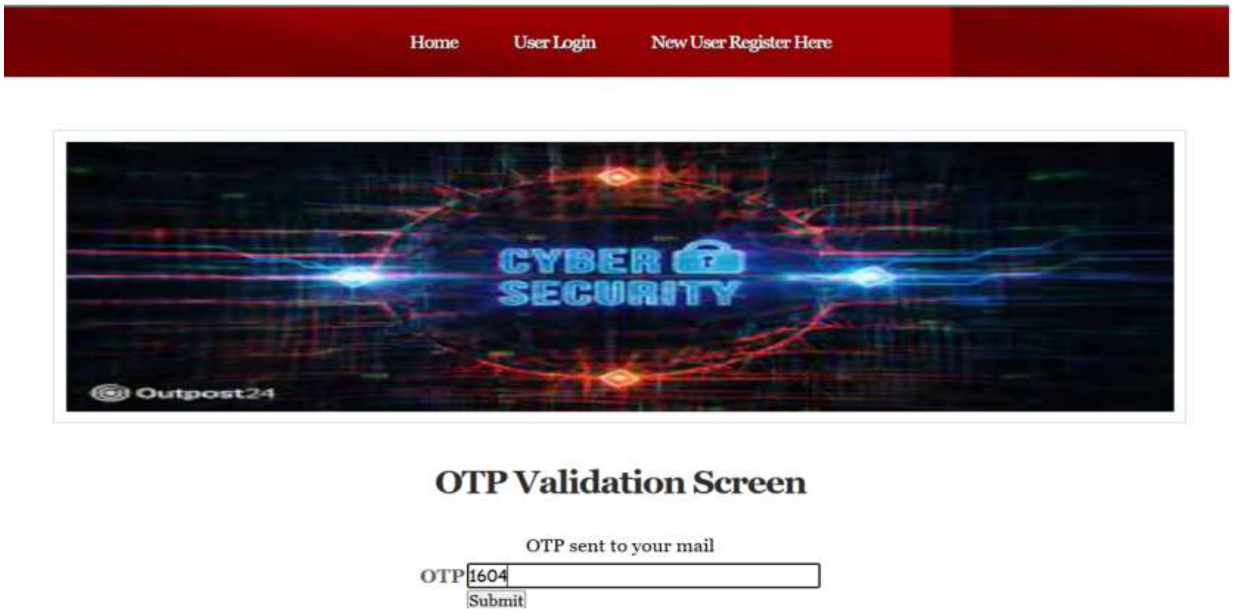


Fig 5 : Enter the OTP and submit

This figure displays the OTP entry screen, where users enter the one-time password received via email and submit it for verification and secure access.

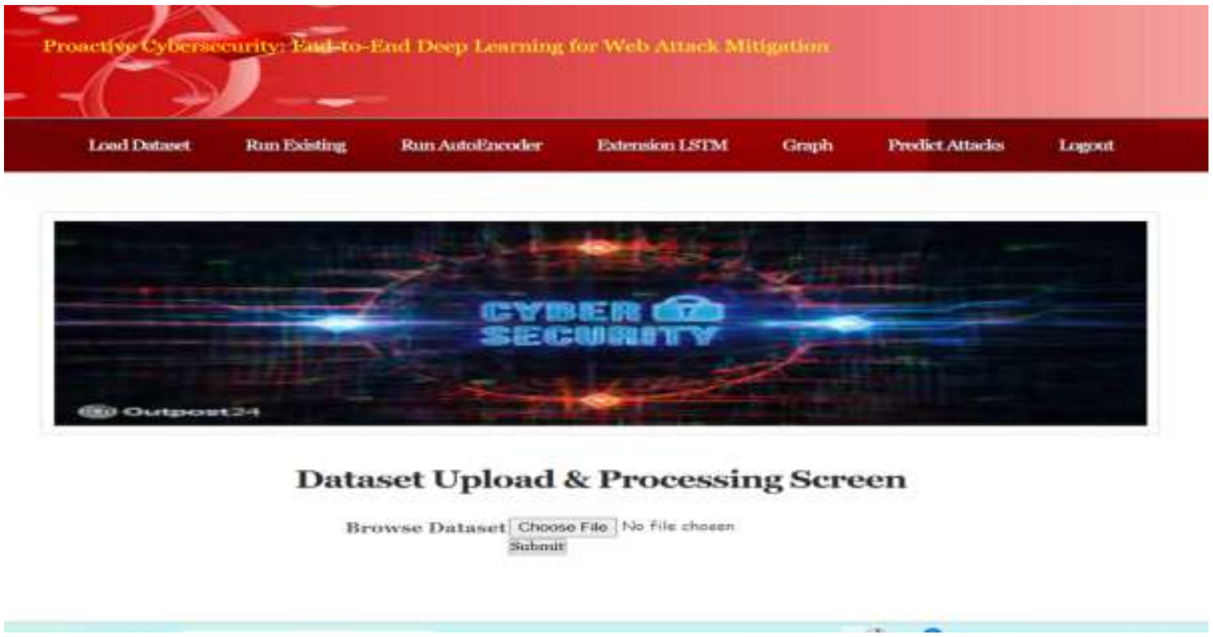


Fig 6 : Upload Dataset

This figure displays the dataset upload interface, allowing users to select and upload files for processing, analysis, or model training within the application.

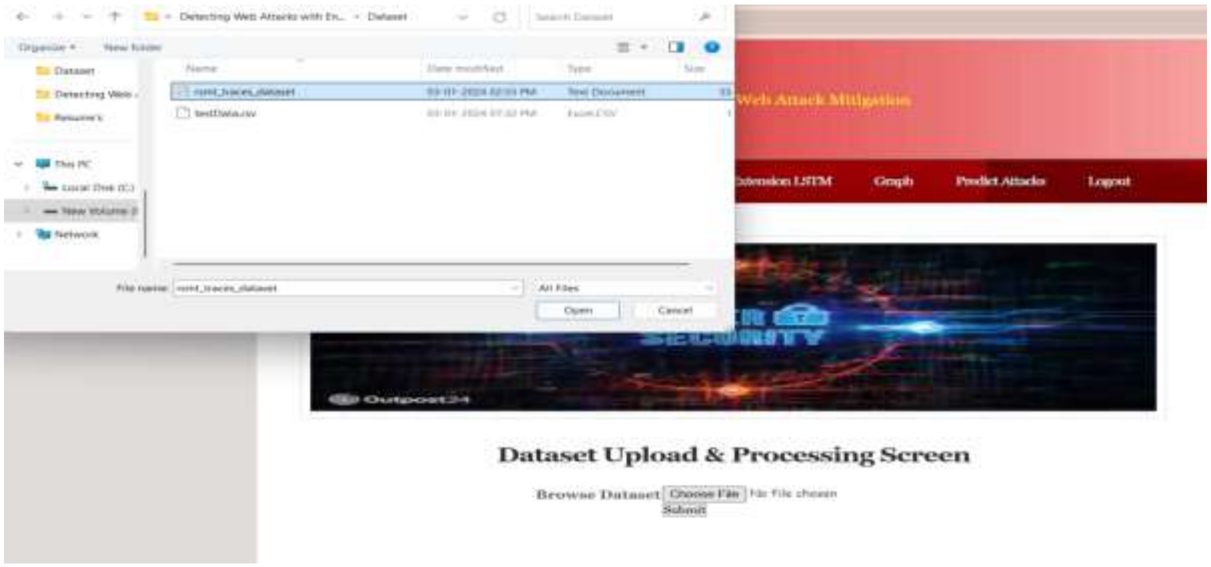
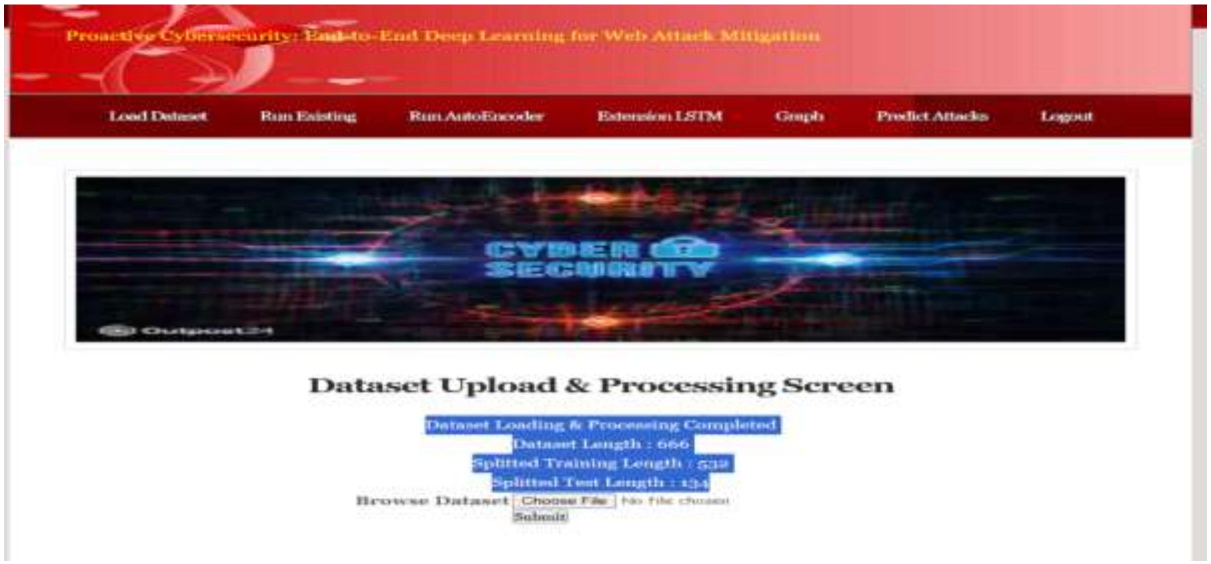


Fig 7: Data set upload and open

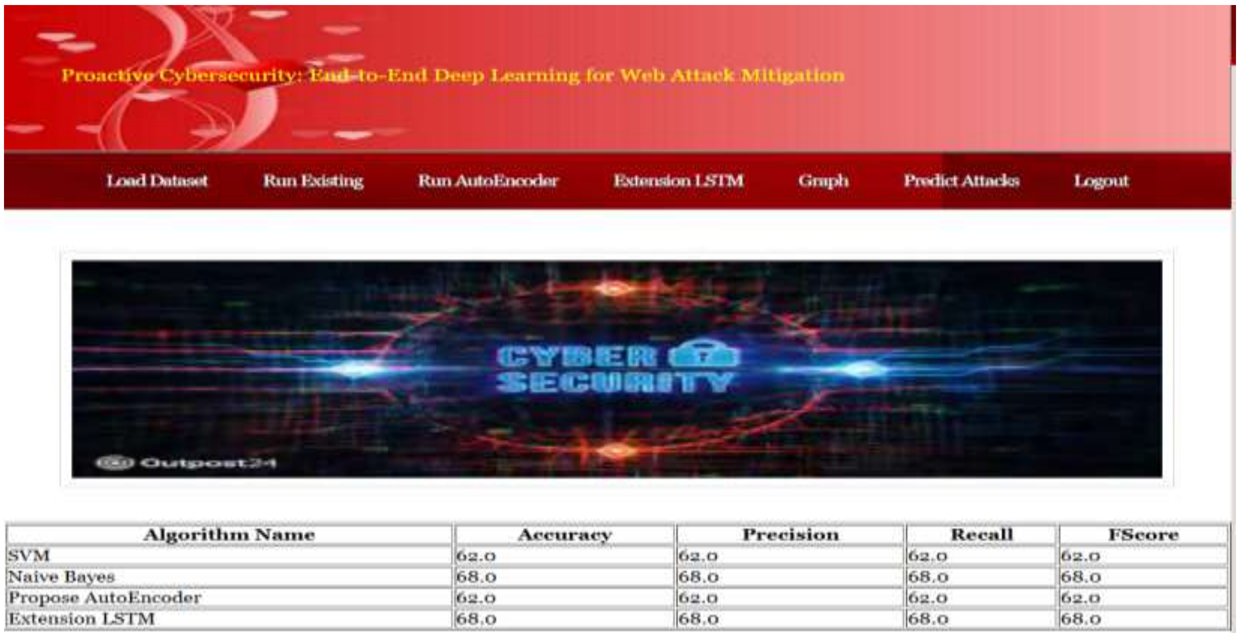
This figure displays the interface where users can upload a dataset and open it for processing, analysis, or model training within the application.



Dataset Loading & Processing Completed
Dataset Length : 666
Splitted Training Length : 532
Splitted Test Length : 134

Fig 8 :Dataset loaded and processed

This figure shows the successful loading and processing of the uploaded dataset, making it ready for analysis, model training, or further operations.



Algorithm Name	Accuracy	Precision	Recall	FScore
SVM	62.0	62.0	62.0	62.0
Naive Bayes	68.0	68.0	68.0	68.0
Propose AutoEncoder	62.0	62.0	62.0	62.0
Extension LSTM	68.0	68.0	68.0	68.0

Fig 9 : Extension LSTM

This figure displays the implementation of the Extended LSTM model, enhancing sequential data analysis for improved predictions and anomaly detection in the dataset.

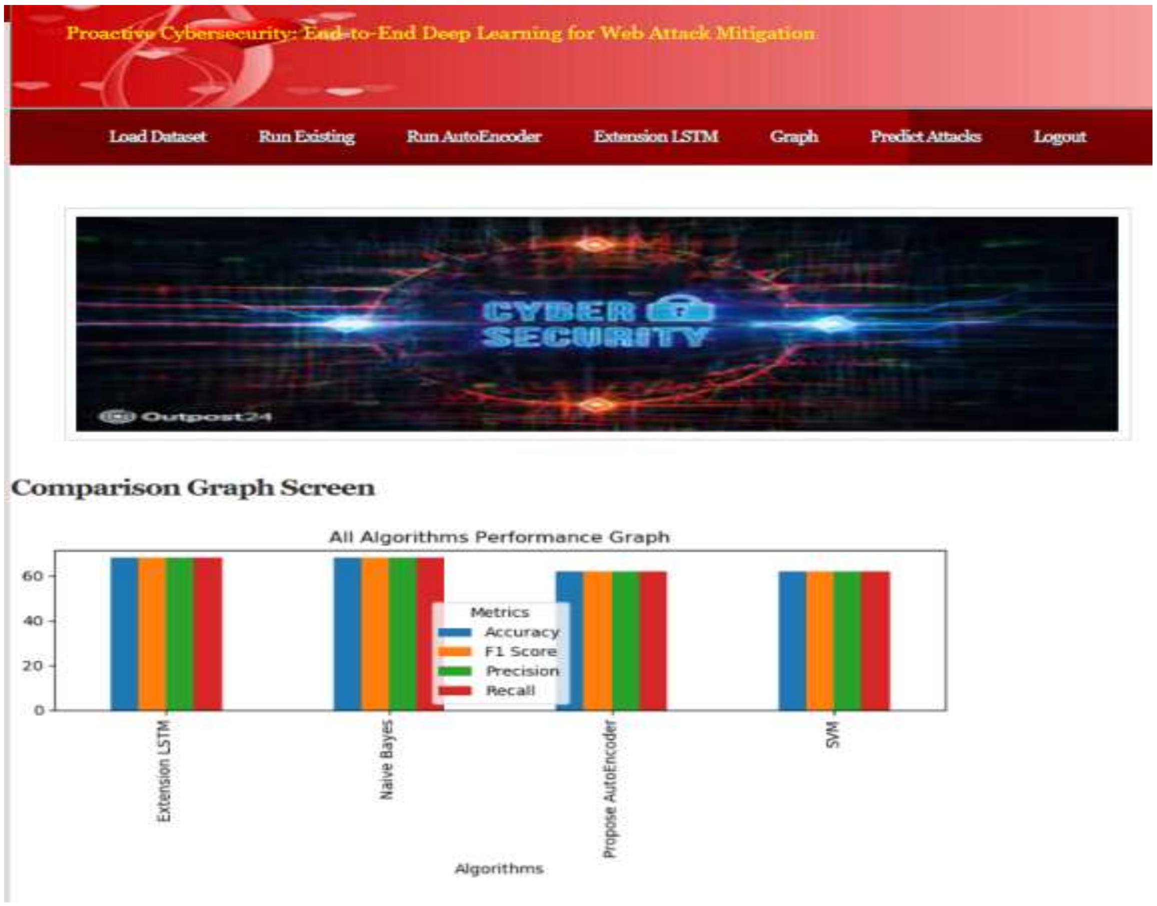


Fig 10 : Comparison Graph
This figure displays a graphical comparison of model performances, showcasing accuracy, loss, or other key metrics for evaluating different algorithms used in the system.

CONCLUSION

As cyber threats continue to evolve in sophistication, traditional security elements like rule-based and signature-based discovery systems are proving insufficient in protecting web applications. These systems rely on pre-defined attack patterns and need frequent manual updates to detect contemporary threats. Either way, they suffer from several drawbacks, including high false positive and incorrect negative rates, limited flexibility, and an inability to examine multi-dimensional attack vectors. The growing modernity of cybercriminals demands a more astute and proactive response to web security. The model framework introduces end-to-end deep learning as an innovative setup for web attack mitigation. Utilizing deep learning patterns, the model framework can, therefore, adapt and learn on evolving threats independently without relying on physically created markings or rules. This enables known and already hidden attack patterns to be identified by the framework in real time, providing a better robust and defensive tool against cyber security threats like SQL injection, cross-site scripting, and distributed denial-of-service attacks. One of the most important characteristics of this methodology is its real-time location and response ability. Far from common frameworks that need periodic overhauls, the deep learning display continuously learns from contemporary assault data and auto-tunes itself to evolving threats. Furthermore, the incorporation of multi-modal analysis scanning HTTP headers, ask payloads, and client conduct improves the accuracy of threat discovery while altogether reducing false alarms. By providing an adaptable, agile, and robotized security system, the suggested profound learning-based arrangement presents a cutting-edge cyber security setup that can secure web applications against the dynamic landscape of cyber threats. This research constitutes a key milestone toward astute, self-developing web security, providing higher levels of assurance, unshakeable quality, and adaptability for today's advanced foundations.

References

1. Acharya, A., & Bhalja, B. R. (2024). Deep Learning-based Detection and Mitigation Strategy for Cyber-attacks on Advanced Metering Infrastructure.
2. Xia, X., et al. (2023). ETD-CV-LSTM: Deep Learning Approach to Electricity Theft Detection in Smart Grids.
3. Abdulaal, M. J., et al. (2022). Real-Time Detection of False Readings in Smart Grid AMI Using Deep and Ensemble Learning.
4. Takiddin, A., et al. (2022). Deep Autoencoder-Based Detection of Electricity Theft Cyberattacks in Smart Grids.
5. Takiddin, A., Ismail, M., Zafar, U., & Serpedin, E. (2022). Deep autoencoder-based anomaly detection of electricity theft cyberattacks in smart grids. IEEE Systems Journal, 16(3), 4106-4117.
6. Qi, R., Zheng, J., Luo, Z., & Li, Q. (2022). A novel unsupervised data-driven method for electricity theft detection in AMI using observer meters. IEEE Transactions on Instrumentation and Measurement, 71, 1-10.
7. Ünal, F., Almalaq, A., Ekici, S., & Glauner, P. (2021). Big data-driven detection of false data injection attacks in smart meters. IEEE Access, 9, 144,313-144,326.
8. Ismail, M., Shaaban, M. F., Naidu, M., & Serpedin, E. (2020). Deep learning detection of electricity theft cyber-attacks in renewable distributed generation. IEEE Transactions on Smart Grid, 11(4), 3428-3437.
9. Bin, Q., Ziwen, C., Yong, X., Liang, H., & Sheng, S. (2020). Rogue base stations detection for advanced metering infrastructure based on signal strength clustering. IEEE Access, 8, 158,798-158,805.
10. Amara Korba, A., Tamani, N., Ghamri-Doudane, Y., & Karabadji, N. E. I. (2020). Anomaly-based framework for detecting power overloading cyberattacks in smart grid AMI. Computers & Security, 96, 101896.
11. Sun, C.-C., Cardenas, D. J. Sebastian, Hahn, A., & Liu, C.-C. (2021). Intrusion detection for cybersecurity of smart meters. IEEE Transactions on Smart Grid, 12(1), 612-622.

12. Wei, L., Rondon, L. P., Moghadasi, A., & Sarwat, A. I. (2018). Review of cyber-physical attacks and counter defense mechanisms for advanced metering infrastructure in smart grid. 2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D), 1-9.
13. Peng, Y., Yang, Y., Xu, Y., Xue, Y., Song, R., Kang, J., et al. (2021). Electricity theft detection in AMI based on clustering and local outlier factor. IEEE Access, 9, 107,250-107,259
14. M. Suresh Babu, Pranaynath Reddy.A Privacy Preservation in Dynamic Data Through Synonymous Linkage on Micro Aggregation ISBN: 978-1-032-66547-4 (pbk) Tylor & Francis -CRC Press – Jan 2024 ISBN: 978-1-032-66553-5
15. M. Suresh Babu - Cryptographic services to enforce secure messaging and data storage - International Journal of Innovative Science and Research Technology - ISSN - 2456- 2165- Nov 2022
16. Halfond WG, Viegas J, Orso A. A classification of sql-injection attacks and countermeasures. In: Proceedings of the IEEE International Symposium on Secure Software Engineering. IEEE; 2006. p. 13–5.