Culturally Adaptive Neural Network for Detecting Cybersecurity Vulnerabilities in Bangladeshi Web Applications

by

Mohosina Shaolin
22101742
Fariha Nawar
22101827
Arik Ahmed Siddique
22101023
Shaikh Mohammad Ali Shams
22101614
Nafisa Maliyat
21301362

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
January 2026.

© 2026. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Mohosina Shaolin

22101742

Fariha Nawa

22101827

Arik Ahmed Siddique

22101023

Shaikh Mohammad Ali Shams

22101614

Nafisa Maliyat

21301362

Approval

The thesis titled "Culturally Adaptive Neural Network for Detecting Cybersecurity Vulnerabilities in Bangladeshi Web Applications" submitted by

1. Mohosina Shaolin (22101742)
2. Fariha Nawar (22101827)
3. Arik Ahmed Siddique (22101023)
4. Shaikh Mohammad Ali Shams (22101614)
5. Nafisa Maliyat (21301362)

of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in January 31, 2026.

Examining Committee:

Supervisor:
(Member)

Moin Mostakim

Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Abdullah Al Nakib

Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

As cyber threats become more complex and frequent, conventional methods for detecting website vulnerabilities, such as rule-based and heuristic approaches, faces significant difficulties, including limited adaptability, high rates of false positives, and a lack of contextual insight. This study presents a predictive model based on neural networks aimed to actively evaluating website security. By applying essential features like security headers, SSL/TLS settings, and SQL injection vulnerabilities, the model detects complex patterns and irregularities, enabling precise identification of emerging threats and vulnerabilities. This approach uses data-driven feature engineering and training with custom neural architectures, for comparison we used random forest and gradient boosting, For explainability we used SHAP followed by evaluation metrics such as precision, recall, and F1-score. Key results show improved accuracy, reduction of false positives, automated monitoring of configurations, and enhancement of resilience against adversarial attacks. Although neural networks show significant potential for transformation, challenges related to transparency, computational demands, and data imbalance are acknowledged. This highlights the necessity for ongoing learning, scalability, and integration with current frameworks, laying the groundwork for robust and adaptable web security strategies.

Keywords: Neural Networks, Website Vulnerability Detection, Cybersecurity, Web Security, Machine Learning, Artificial Intelligence, Rule-based Security Systems, Heuristic Methods, Zero-day Vulnerabilities, Security Headers, SSL/TLS Configurations, SQL Injection, Adversarial Attacks, Feature Engineering, Anomaly Detection, Explainable AI (XAI), Proactive Security Solutions

Acknowledgement

We would like to express our deepest gratitude to our supervisor, Moin Mostakim, and our co-supervisor, Abdullah Al Nakib, for their valuable guidance, encouragement, and insightful feedback throughout our research. We also extend our sincere appreciation to BRAC University for providing us with the necessary resources and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Web application vulnerabilities are security weaknesses which attackers exploit to break into websites and steal information and disrupt computer systems. The coding practices and configuration flaws and a lack of security measures create these vulnerabilities. The most common security threats include injection attacks and cross-site scripting and weak authentication and insecure data transmission.
Machine learning enables computers to learn from data and make decisions without following fixed rules. Machine learning allows cybersecurity systems to analyze extensive data sets and identify patterns that may cause potential security threats. Machine learning provides a better solution than traditional rule-based methods because it helps the systems to handle new and unknown security threats.
Neural networks are machine learning models that work in a similar way to the human brain. They learn to recognize intricate patterns by using their multiple data processing layers. Neural networks have proven to be more effective when handling extensive datasets that have a variety of data including web application security information. Neural networks are used to extract patterns from both vulnerability data and website operational properties.
The preparation process needs to be completed before data can be used in neural networks. Websites contain various features which include backend technology and framework and location information that exist as non-numeric data. The model requires numeric conversion of these features to enable its learning process. The techniques of encoding and grouping enable clear representation of this information. Also, UMAP and t-SNE methods are used to show the relationship between data points through their visualization techniques. These methods transform complex data into simplified visual representations which is helpful to understand website patterns and clusters and their matching similarities.

1.2 Problem Statement

Web application vulnerabilities are security weaknesses which attackers exploit to break into websites and steal information and disrupt computer systems. The coding practices and configuration flaws and a lack of security measures create these vulnerabilities. The most common security threats include injection attacks and cross-site scripting and weak authentication and insecure data transmission.

Machine learning enables computers to learn from data and make decisions without following fixed rules. Machine learning allows cybersecurity systems to analyze extensive data sets and identify patterns that may cause potential security threats. Machine learning provides a better solution than traditional rule-based methods because it helps the systems to handle new and unknown security threats.
Neural networks are machine learning models that work in a similar way to the human brain. They learn to recognize intricate patterns by using their multiple data processing layers. Neural networks have proven to be more effective when handling extensive datasets that have a variety of data including web application security information. Neural networks are used to extract patterns from both vulnerability data and website operational properties.
The preparation process needs to be completed before data can be used in neural networks. Websites contain various features which include backend technology and framework and location information that exist as non-numeric data. The model requires numeric conversion of these features to enable its learning process. The techniques of encoding and grouping enable clear representation of this information. Also, UMAP and t-SNE methods are used to show the relationship between data points through their visualization techniques. These methods transform complex data into simplified visual representations which is helpful to understand website patterns and clusters and their matching similarities.

1.3 Objective

1. **Develop an Intelligent Vulnerability Detection System:** Develop an intelligent system that is able to detect changes in websites and evaluate their vulnerability status based on neural network-based predictions.

2. **Proactive Cyber Threat Response:** As the era of cyber threats is evolving rapidly, it's crucial to develop proactive techniques to identify potential weaknesses and vulnerabilities in web applications.

3. **Security Feature Extraction:** Extract security-related features from websites, such as:

   - Security headers
   - SSL/TLS configuration
   - SQL injection vulnerabilities

   These features are key indicators for the overall state of a physical website's defense.

4. **Continuous Monitoring:** Monitoring these features continuously makes it much easier to find malicious changes.

5. **Deep Pattern Recognition Using Neural Networks:** Deploy neural network-based models to:

   - Identify complex and hard patterns within the features
   - Predict vulnerabilities with very high accuracy

6. **Automated Vulnerability Evaluation:** The proposed system aims to automate the vulnerability evaluation process. Ensures real-time detection and immediate response to security issues.

7. **Reduced Human Effort and Error:** This automation helps to:

   - Greatly reduce human effort
   - Minimize human error in vulnerability detection

8. **Real-time Security and Prediction:** Specifically extracts and monitors features like:

   - Security headers
   - SSL/TLS configuration
   - SQL injection vulnerabilities

   Neural networks enable the system to:

   - Identify deeper patterns
   - Forecast security vulnerabilities with extreme accuracy

9. **Minimize Detection Delay:** The automated process ensures vulnerabilities are not delayed in being detected and responded to.

10. **Strengthen Organizational Cyber Defense:** Helps organizations:

    - Confront potential cyber threats
    - Fortify their web security
    - Eliminate the chances of exploitation by attackers

Chapter 2

Literature Review

2.1 Review of Existing Research

As the number of cyber-attacks on sites and web applications continues to surge, the importance of cyber security measures have steadily increased. New vulnerabilities are being explored utilizing neural networks as conventional methods are unable to cope with these sophisticated attack techniques.
For example, in [23], an AI-based security integration and assessment framework was developed with the help of supervised learning methods. Likewise, [25] focused on the identification of cyber attacks in Bangladesh using machine learning techniques, whereby [37] effectively illustrated that artificial neural networks reduce false positives while detecting threats.
Other sources involve [28], which developed and proposed the automatic vulnerability detection system using CNNs and [17] which focused on vulnerability detection in Bangladeshi websites. Furthermore, [21] began the exploration of a hybrid graph neural network for the identification of PHP vulnerabilities, along with [12], which created a neural network prediction model to gauge software vulnerabilities.
With [35] implementing CodeBERT alongside the CNN model, the results exhibited a great improvement in automated software vulnerabiltiy detection compared to basic neural network models. While [39] began the adoption of AI based systems with the help of neural networks to identify problems before the arise.
In their work, [15] developed a behavioral analysis tool called AppMine that tracks web application vulnerabilities. Their approach involved using behavioral analysis and machine learning algorithms for threat identification during active web sessions. In particular, [26] summarized the detection of vulnerabilities with the aid of deep learning, in particular focusing on the use of CNNs for malware detection and RNNs for intrusion detection systems.
AI-enhanced penetration testing techniques for security assessments were researched by [24], who showcased a combination of AI with other technologies as an essential idea of modern warfare. [27] examined a cybersecurity threat study in Bangladesh and proposed AI-based technologies as interventions for the susceptible issues raised. Moreover, other works have also added important contributions addressing AI-based solutions for cybersecurity issues. Adaptive neural networks have been proposed for dynamic attack detection in [34], in which they claimed that the classification accuracy improves under real-time conditions. [16] proposed deep reinforcement learning models to be used in the realm of cybersecurity where adapting to new

types of evolving cyber threats will be a necessity. Another model based on a memristor with an improved TCNN structure to enhance threat prediction was put forward in [41]. The advantages of deep neural networks over shallow models for intrusions detection were put on display by [14] on environments with advanced threats from adversaries. Multilingual deep learning algorithms for cyber threat intelligence focused on cross linguistic detection of threats was examined by [18]. AI measures for securing online transactions are exposed in [30] where security measures for e-commerce platform transactions were proposed.

Cyber-physical systems (CPS) has become increasingly challenging in terms of cybersecurity due to the rise of interconnectivity and digital dependencies. According to a systematic review by Sagar et al., deep neural networks (DNN) are being studied for vulnerability detection in CPS as it offers adaptive learning methods from attack tactics and anomaly detection capabilities [38].

Machine learning (ML) techniques such as Random Forest have shown prospective results in cyber attack detection. A study by Oyetoro et al. explores these techniques to detect and classify network anomalies as it reports a high accuracy rate in intrusion detection scenarios [31]. Similarly, logistic regression is also a widely used statistical model for malware detection due to its interpretability and efficiency, as shown by Akram et al. [22].

Singh has developed an ML based vulnerability detection system, in the context of web applications , using Random Forest classifiers that effectively identify common exploits with high precision [43]. Unsupervised algorithms like K-Means clustering have also been utilized for anomaly detection in cybersecurity, which provide flexible and lightweight clustering for behavior analysis. [33].

Multiple analytic studies comparing them stated that even though DNNs and ensemble learning models provide high accuracy, their training complexity and data dependency is a major concern. A research done by Tanko Bakar. and Alzaabi et al. highlights that the success of ML algorithms depends significantly on dataset quality, feature engineering, and attack type [32], [42].

Recent developments include hybrid systems combining statistical, heuristic, and ML-based methods to improve real-time detection in high-load environments [36], [40]..

2.2 Summary of Key Findings

The previous studies have shown that traditional Rulebased security methods fail to protect modern web systems because they cannot identify new security threats because of advanced vulnerabilities. In these scenarios, machine learning methods especially, neural networks improves the performance because the systems acquire knowledge from data while they learn to recognize new attack patterns.

The research also shows that data quality along with proper preprocessing methods will determine how well models perform their tasks. The process of grouping similar security vulnerabilities together with the conversion of categorical data into numerical values will assist models to identify security patterns using better pattern recognition.

The website security patterns of different regions evolve according to their available technological resources because different regions implement distinct security methods. This research studies culturally adaptive systems that can learn specific

patterns used in Bangladeshi web applications yet still works with international websites. The research findings support the methods used in this study which uses neural networks together with data preprocessing methods and visualization techniques.

Chapter 3

Requirements, Impacts and Constraints

3.1 Societal Impact

This research makes a contribution towards enhancing awareness on cybersecurity practice because it emphasizes the possibility of development behavior being reflected through recurring vulnerability patterns. Examining the security vulnerabilities that are culturally sensitive, the study helps towards improved insights into the software security risks in the web application. The results can be used by developers, organizations, and researchers to advance the concept of secure coding practices and enhance the overall web applications safety without reaching out to individuals or communities.

3.2 Environmental Impact

There was no form of environmental interference because the entire project was carried out in a digital environment. There was no use of physical resources or materials other than normal computing equipment. Although the training and testing of the neural network were done using computational resources.All tests were done on a regular computing platform that the researchers had access to without the involvement of high performance or massive computing infrastructure. Unnecessary computational overhead was kept to a minimum by utilizing trained models whenever feasible and optimization of experimental computations, hence responsible and efficient use of energy in the course of research.

3.3 Ethical Issues

This research was conducted in compliance with standard academic and data ethics guidelines.The information incorporated in this study was gathered from publicly accessible sources . GitHub repositories visited were only those available publicly, without verifying the existence of developer related location information in any of them or retrieving any data that could be classified as private, restricted, or personally sensitive. On the same note, information and metadata of websites were also gathered in publicly available sources of registered sources in Bangladesh, where

the information is publicly available. No information was gathered by means of unauthorized access, scraping on personal systems, or breach of platform terms of service.
This study also did not have the purpose of targeting or identifying individual developers or organizations. This was analyzed on a pattern and dataset scale and based on the aggregated vulnerabilities features, and not on individual attribution. The findings are only provided in academic and analytical use and they are aimed at promoting the study of cybersecurity and not profiling or name calling. This study has ensured that there was integrity in the data collection, analysis, and reporting processes, because it relied on the source of public data and did not involve any personal identification.

3.4 Project Management Plan

The implementation of the research was done in organized phases to achieve timely and effective implementation. The preliminary stage was literature review and problem definition, which was followed by collecting data since publicly available sources were available. The second stage was the design of the neural network, its training, and evaluation. The last step was to carry out result analysis, documentation and thesis preparation. Frequent progress review made sure that every stage was achieved within the scheduled plan.

3.5 Risk Management

There are a number of risks that were taken into consideration with the research. The risk regarding data was also reduced to the extent of using the publicly accessible and validated sources. Technical risks like overfitting of model or wrong predictions were checked by thorough training and evaluation. The risk is reduced through ethics since it did not identify any personal information but only general trends of vulnerability. The constant review also minimized the effects of unexpected problems.

3.6 Economic Analysis

This research was performed at a low financial price since it used open source software, data that was available in public access, and computing facilities that were already available to the researcher. No extra hardware or software was needed to specialize in it. This fact is evidenced by the low cost nature of the project, which indicates that culturally adaptive cybersecurity research can be conducted at a lower efficiency and cost, and therefore affordable to academic and research institutions.

Chapter 4

Proposed Methodology

4.1 Design Process or Methodology Overview

The datasets used in this research are : Final Dataset and Dataset 2

4.2 Data Collection & Cleaning

We have collected the website links in a csv file from BASIS(Bangladesh Association of Software and Information Services), the national association of Bangladeshi companies. We collected the foreign websites from a publicly available dataset on Kaggle. It is a reputable and trusted system of high quality datasets for research and academic purposes. This dataset includes information about vulnerabilities and website references. While collecting the website links, we manually verified if there were any invalid or duplicate URl. This csv file that contained unique URLs in a column was the initial input file for analysing vulnerabilities. We needed our dataset to be accurate ,clean and reliable so we preferred manual collection so that we can check properly and avoid incorrect data.

4.2.1 Data Pre-Processing

Vulnerability Detection:

We implemented a web security analysis script for scanning web vulnerabilities and data processing. It is a rule based method. This framework reads the URLs from the file and sends requests to every website using HTTP or HTTPS protocols . This script adds test values to the website links and inspects the existence of security weaknesses. At first,the script injects test inputs like SQL injection, XSS etc and validates the outcome. It looks for sensitive keywords in responses and also determines if there are accesses to restricted paths.The outputs of each function are returned in human readable textual form which is later converted into binary form. This script finds a wide range of vulnerabilities and most dominant and common vulnerabilities are stored in each column. The output values are in binary format. If the returned binary value is 0 then the website is safe from the vulnerability and if the value is 1 then the vulnerability is evident in that website. The outputs are returned in binary format because this format is compatible with machine learning models and the dataset can be clustered.But in textual form it is difficult as

they show discrepancy. This script is the most suitable for this work because it detects vulnerabilities of a large number of sites.This automated method can conduct security analysis which is fair and scalable.

Metadata Extraction

Metadata extracting was the next step of our work. We developed a metadata extraction and web profiling tool. This script analyzes websites by making HTTP/HTTPS requests and retrieves available metadata. After sending the requests a response is obtained and the script retrieves metadata that gives information about the way the website is hosted and programmed. The HTML text of every website is inspected with the help of comparing keys and detecting patterns. It identifies technologies like drupal,angular,react etc and backend programming languages. This script also identifies the server and website category. The server information is extracted from HTTP headers and the website category is determined by analyzing domain extensions. But this script could not extract information from all websites because of domain expiration or the server blocked requests for security purposes. Again some websites hide this information. So, we used a chrome extension named wappalyzer and curated the information manually . For determining the location at first we generated an automated script for discovering github links. This script reads the input data and delivers search queries to Google through the SerpAPI and takes the valid github link as result. The extracted github links that are extracted are stored in a separate csv file which is input for another script generated by us for detecting location. This script identifies github organization names and scans user and company terminal to get location information. Here authentication tokens are enabled to extend API request quotas.For some website links it was difficult to find github link and location for some reasons like API limitation ,private github profiles. So, we had to identify manually in those cases. We put the unknown in the cell where the information was not found.

4.2.2 Data Transformation

One-hot Encoding

One-hot encoding is used in this dataset to transform the categorical information into a numerical form. The dataset contains some descriptive features like backend technology, server type, framework and website category, which was represented as text labels previously. Using the one-hot encoding process, each unique category within these features was converted into a separate binary value (0 or 1). For example, backend technology such as PHP, python were represented as individual columns, where 1 represents that this specific backend technology was used to build this web application and 0 means it was not used.

This encoding process was applied for the categorical attributes for backend technology, server type, framework and website category in the dataset to ensure the clear representation of characteristics of the web applications [29]. By using numerical values instead of string-based values help the neural network learn patterns more accurately. Numerical inputs give a more precise representation during model training. As the dataset contains a diverse mix of the technological landscape of Bangladeshi and foreign websites, one-hot encoding plays an important role in increasing model accuracy by helping the neural network to effectively understand the

categories when detecting website vulnerabilities. The vulnerabilities were grouped based on similar attack mechanisms which targeted specific system components. The approach simplifies the dataset while establishing security patterns that the model can better understand.
The vulnerabilities are categorized in this dataset in a manual rule-based procedure based on a set of standardized vulnerability grouping strategies. Similar attack techniques and exploit patterns are discovered and classified according to their application in attacks [20]. This method is used to explain how the vulnerabilities can be systematized into classes by looking at the causes of the vulnerability, the common features of the vulnerability, and the security presumptions that are violated during an attack [4].Each vulnerability is examined in terms of its category, attack manner ,component that it affects and the fundamental weakness .Based on the instructions provided by these frameworks, the vulnerabilities are classified under one of the six groups which are defined earlier.Subsequently these are also encoded using one-hot encoding which is machine-readable. The grouping itself is manually and this guarantees stability, readability, and agreement with existing literature on vulnerability grouping.

Group A: Injection & Input Attacks

This group includes vulnerabilities caused by improper input validation, allowing attackers to inject malicious code or commands that lead to unauthorized execution and data access.

Group B: File & Path Attacks

This group consists of vulnerabilities related to file access and path manipulation, enabling attackers to read, include, or expose sensitive server files.

Group C: Cross-Site Attacks

Vulnerabilities in this group affect interactions between web applications and user browsers, exploiting user trust to execute scripts or perform unsafe redirections.

Group D: Authentication & Access Control

This group includes vulnerabilities related to identity verification and authorization, which can result in unauthorized access to protected resources.

Group E: API & Request Manipulation

This group includes vulnerabilities resulting from improper handling of API and server-side requests, allowing attackers to misuse internal services or bypass security controls.

Group F: Encryption & TLS Issues

This group contains vulnerabilities related to weak encryption and insecure communication channels, increasing the risk of data interception during transmission.

Vulnerability Grouping Table

Table 4.1: Vulnerability Grouping Based on Attack Mechanisms

Group	Vulnerabilities in this Dataset
Group A	SQL Injection; Injection Problem; Expression Language Injection; CRLF Injection; CSV Injection; XXE Processing; XSS; Improper Data Validation; Deserialization of Untrusted Data; Insecure Deserialization
Group B	LFI; RFI; Unrestricted File Upload; Insecure Temporary File; Unreleased Resource
Group C	CSP; CSRF Protection; Unvalidated Redirects; Information Exposure via Query Strings
Group D	Broken Authentication; Broken Access Control; Sensitive Data Exposure; Hard-Coded Password; Empty String Password; Allowing Domains/Accounts Expire; Insufficient Entropy
Group E	API Security; API Security Vulnerabilities; SSRF; Logging & Monitoring; Security Misconfiguration; Third-Party Components; Vulnerability Scanning Tools; Vulnerability Template; Insecure Third Party Domain Access; Business Logic Vulnerability; Obsolete Methods; Follina Vulnerability
Group F	Risky Cryptographic Algorithm; Heartbleed Bug; Insecure Transport; Insecure Randomness; Insecure Compiler Optimization; Full Trust CLR Verification; Unsafe JNI; Unsafe Mobile Code; Unsafe Signal Handler Call; Unsafe Reflection; Buffer Overflow; Undefined Behavior; Improper Pointer Subtraction; Missing Check against Null; Unchecked Error Condition; Unchecked Return Value; Catch NullPointerException; Covert Storage Channel; Using Freed Memory; Doubly Freeing Memory

For example, SQL Injection and CRLF Injection use unsafe user input to launch dangerous attacks which lead to two different types of attacks: one which attacks databases and one which attacks web headers because both attacks share their fundamental method which uses bad input to create security vulnerabilities. In contrast, Group F contains Insecure Transport security because this security problem does not involve input injection but instead shows data encryption failures that occur during user to server data transmission.

4.2.3 Exploratory Data Analysis (EDA)

UMAP

This figure here represents the Three_Dimensional UMAP (Uniform Manifold Approximation and Projection) visualisation of websites of our dataset based on their Location, Backend Programming Language and Frameworks used. All these features are categorical so they are encoded into numerical values and then projected

into the Lower Dimensional space using UMAP. Each dot here represents one website. The websites are defined by Location, Backend Language and Framework. The close dots represent that the websites use a similar tech stack and share the same location. The far apart dots represent the websites using different tech stack and from different locations. The cluster of Bangladeshi points represent similar Backend Language and frameworks are popular locally. Mixed clusters represent shared technologies.
The three UMAP dimensions (UMAP1, UMAP2, and UMAP3) do not correspond to specific original features; rather, they are latent dimensions constructed by the algorithm to preserve local neighborhood relationships in the data.

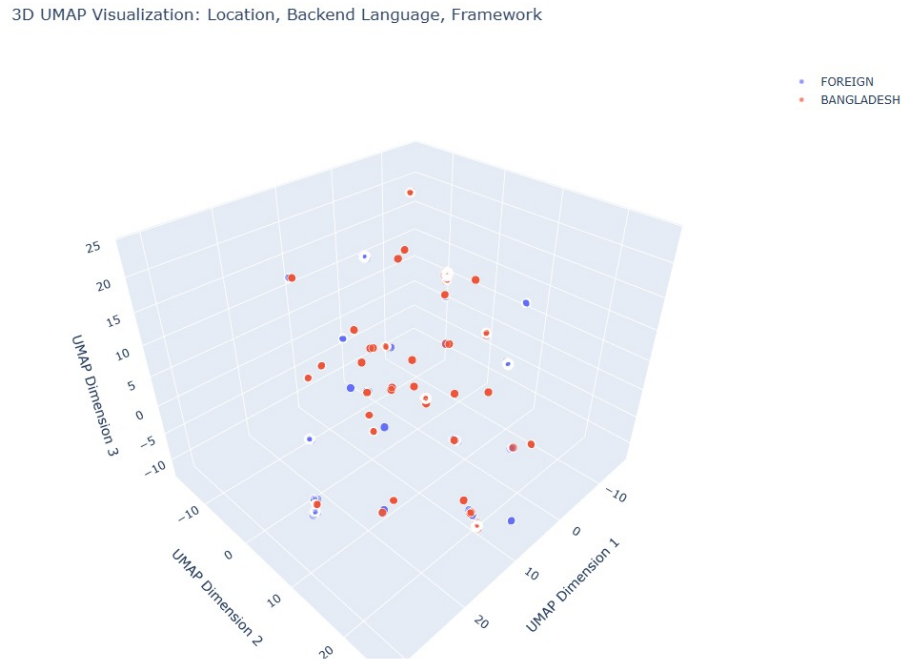

Figure 4.1: UMAP visualization of the processed dataset

t-SNE

This figure presents a 2D tSNE (tDistributed Stochastic Neighbor Embedding) visualisation of our Dataset, where each point represents a single website characterized by two categorical values: Location and Backend Programming Language. The colors represent different backend languages used in different web apps. Backend Languages with similar usage tend to cluster closer that forms a cluster and backend techs with different characteristics appear in distance. The overlapping clusters suggest that the backend language is commonly used in different locations. This 2-dimensional t-SNE visualization is imposed to analyze similarity patterns among websites based on the location and Backend Programming language coloured by Location (Bangladesh, Foreign). The tech stacks using similar characteristics cluster together. Each datapoint here represents one website. The Bangladeshi origin websites are coloured in Yellow and foreign websites are coloured in Purple.

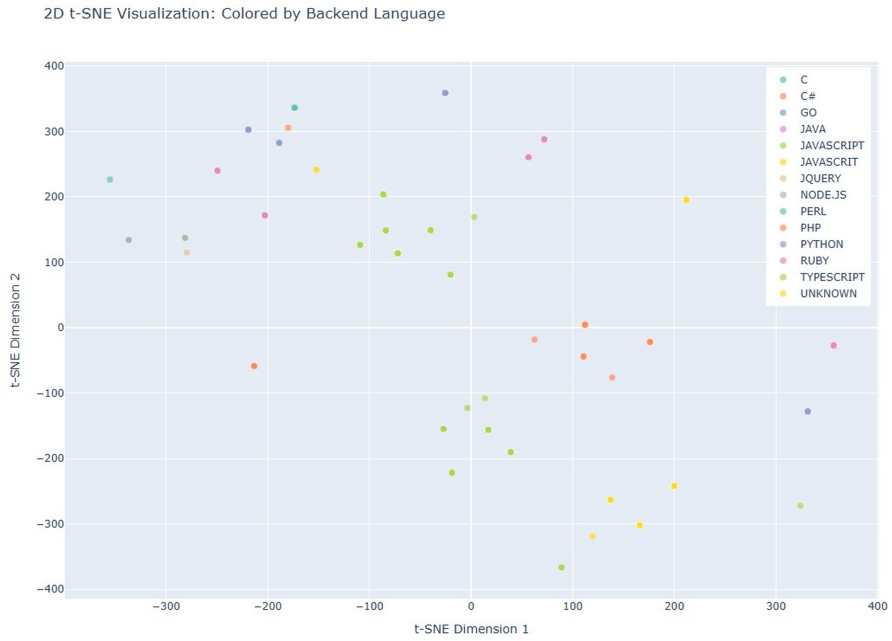

Figure 4.2: t-SNE visualization

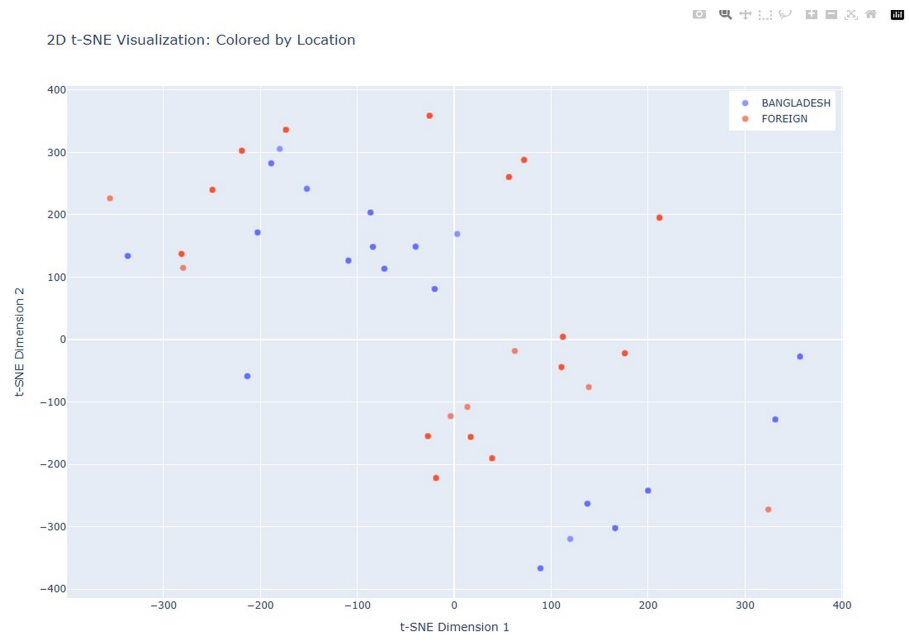

Figure 4.3: t-SNE visualization 2

t-SNE is a method of reducing high dimensional data to a two-dimensional space in a manner that the relative similarity of the data points is preserved. In tsne the common samples are clustered together and the different ones are expanded in the new space.The websites that have similar framework features will be located in closer groups as compared to those that have different framework features which are more widely spaced in this visualization. The color difference by place allows a comparative study of Bangladeshi and foreign websites with respect to both areas of differences and similarities. Consequently, tSNE gives a fair visual representation

of latent structural patterns in the data that cannot be seen with numerical analysis alone.

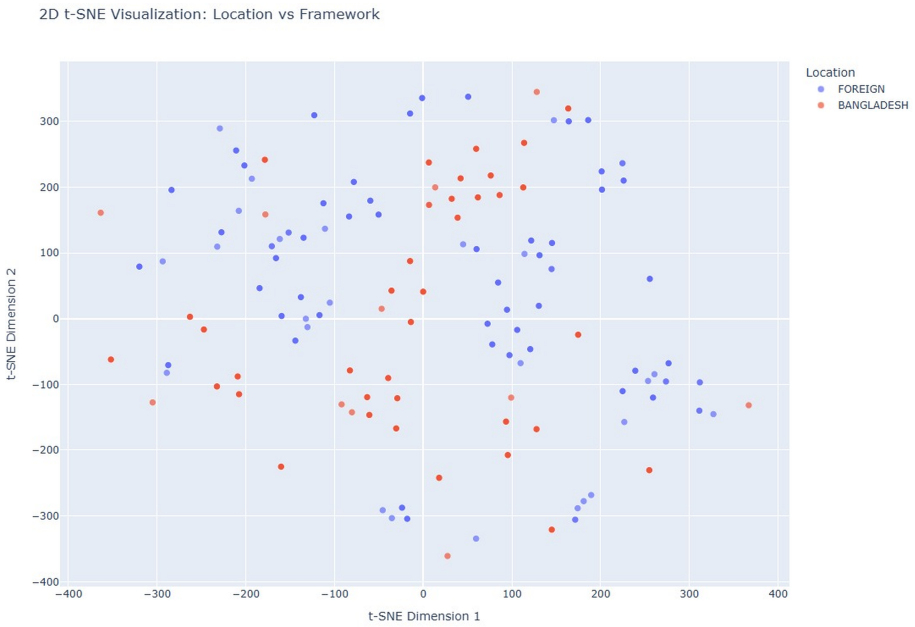

Figure 4.4: t-SNE visualization 3

LCS

The algorithm that was used in this research is the Longest Common Subsequence (LCS) to compare the vulnerability patterns with the pattern of the consolidated dataset that includes both Bangladeshi websites and foreign websites. The vulnerability patterns of these websites were placed in different columns in the dataset, thus making it easier to do a direct comparative analysis of patterns within the two groups. The main aim of using LCS was to find out whether there were any common sub sequences in the two sets of vulnerabilities patterns or not, which in turn could give information about the common weaknesses or possible overlaps in web security between local and international sites.

The reason why LCS has been chosen in the present study is the fact that it is a very effective tool in identifying common patterns among various datasets, even when those datasets do not necessarily match but have certain structural similarities. The LCS algorithm is concentrated on the identification of the longest sequence that is present in both datasets but retains order but has intervening gaps. This feature is especially beneficial in the framework of vulnerability tendencies, where the similarities might occur despite the difference in web architecture.

The main purpose of using LCS was not only to determine the common occurrence of the common patterns but also to examine the possibility that common patterns could represent a common source or similarity between Bangladeshi and foreign websites. A possible conclusion of finding shared patterns is the possible presence of similar developers doing the websites, or that there is similar practice or security weakness in the websites. Such conclusions were supposed to be supported with the help of the LCS analysis because the observation of such common subsequences could indicate that the same developers might have worked on both the Bangladeshi and foreign sites thus creating common patterns.

The study was initiated by the process of a combined data set, which was the combination of the vulnerability trends of both the Bangladeshi and foreign websites. These trends were coded in binary strings whereby a 1 is generally used to represent the occurrence of a vulnerability, and 0 would mean that there is none. Each dataset was then divided into smaller pieces in order to be able to compare the vulnerability patterns in detail. The sizes of the chunks were varied (e.g. 2, 4, 6, 8, etc.), with this granularity used to represent common subsequences at varying degrees of detail. Smaller chunk sizes, e.g. 2 or 4, focused on finding more similarities in the datasets, whereas larger chunk sizes, e.g. 6 or 8, found more general patterns and discovered some more important, and less frequent similarities. This technique gave an all-inclusive perspective of the connections in the vulnerability patterns in the Bangladeshi and foreign sites.

Example 1:Obhai (Bangladeshi) and Ubuntu (Foreign) In the former, the Obhai (Bangladeshi) site and the Ubuntu (foreign) site were taken to a test. Comparison of binary strings in these websites was done in chunks. As an example, the LCS algorithm with a chunk size of 4 detected the common sub-sequences to include:

[h] `[1000][0001]` (Bangladeshi) and `[1000][0001]` (Foreign)
`[0000][0000]` (Bangladeshi) and `[0000][0000]` (Foreign)

These sub sequences are typical patterns of which a vulnerability (1) appears at the same relative positions in both websites, but the non-occurrence of a vulnerability (0) also occurs. The fact that the similar patterns are present in both sets of data and especially in similar locations might indicate that both websites have a set of vulnerabilities, and that they have similar technologies. The increase in the chunk size to 30, however, did not produce any common fractions, and they suggest that large chunks blurrier details and do not show smaller and more specific shared weaknesses.

Pattern Division (Chunk Size = 2)

BD Bangladesh Chunks	Foreign Chunks
[01] [10] [00]	[01] [10] [00]
[00] [00]	[00] [00]
[01] [10] [00]	[01] [10] [00]
[00] [00]	[00] [00]
[00] [00] [00]	[00] [00] [00]
[00] [10]	[00] [10]
[00] [00] [00]	[00] [00] [00]
[00] [00]	[00] [00]
[00] [00] [00]	[00] [00] [00]
[00] [00]	[00] [00]
[00] [00] [00]	[00] [00] [00]
[00]	[00]
Total: 29 chunks	**Total: 29 chunks**

BD Bangladesh Website vs Foreign Website Comparison

Category	BD Bangladesh Website	Foreign Website
URL	https://www.obhai.com/	http://www.ubuntu.com/u
Full Pattern (58 digits)	10000011001000000000100000 00000000001000000000000 0000000000000000	10010011100100000000010000 00000000000010000000000 0000000000000000

Figure 4.5: Comparison between Bangladeshi and foreign websites

Example 2: Link3 (Bangladeshi) as compared to Ubuntu (Foreign). In the second example, there was a comparison between the Ubuntu (foreign) and the Link3 (Bangladeshi) websites. With a chunk size of 2, the LCS algorithm was again used to find common subsets, including:

`[10][00]` (Bangladeshi) and `[10][00]` (Foreign)
`[00][01]` (Bangladeshi) and `[00][01]` (Foreign)

These trends also help support the fact that there are similarities between the Bangladeshi and foreign websites since both datasets showed the same subsequences at smaller chunk sizes. Similar to the previous case, as the size of the chunk to be examined was expanded and common pattern pairs were found.

Figure 4.6: Pattern division comparison between Bangladeshi and foreign websites 2

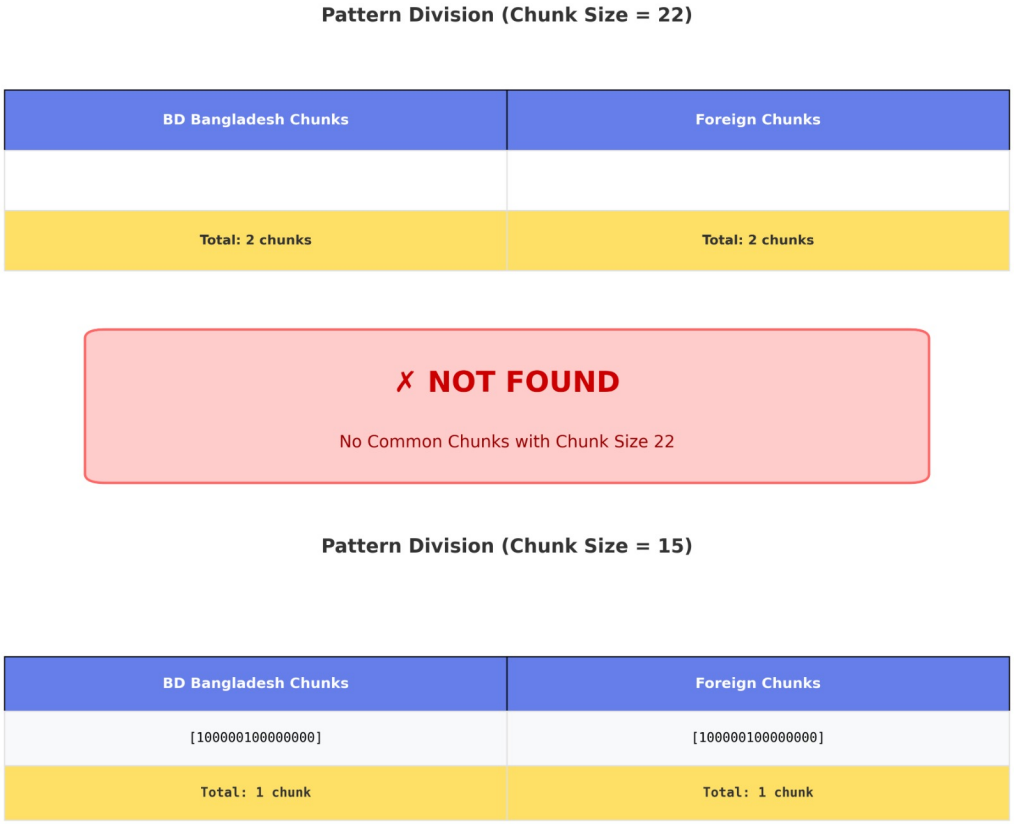

Figure 4.8: Comparison between Bangladeshi and foreign websites

Select Websites to Compare

Category	Website URL
BD Bangladesh Website	https://www.link3.net/
Foreign Website	http://www.ubuntu.com/us...

Pattern Division (Chunk Size = 2)

BD Bangladesh Chunks	Foreign Chunks
[10] [00] [00]	[10] [01] [00]
[00] [11]	[11] [10]
[01] [00] [00]	[01] [00] [00]
[00] [00]	[00] [00]
[10] [00] [00]	[10] [00] [00]
[00] [00]	[00] [00]
[00] [00] [00]	[00] [00] [00]
[00] [00]	[00] [00]
[00] [00] [00]	[00] [00] [00]
[00] [00]	[00] [00]
[00] [00] [00]	[00] [00] [00]
[00]	[00]
Total: 29 chunks	**Total: 29 chunks**

Figure 4.7: Comparison between Bangladeshi and foreign websites

The research was able to provide the commonalities in the vulnerability patterns of the Bangladeshi and foreign websites by applying the LCS algorithm to the merged dataset to identify its commonalities. The results indicate that despite the existence of significant differences, there are some common vulnerabilities, which can be the result of similar development practices or even a common origin. The comparison of the data sets was also subtle with the LCS analysis and especially with the different size of the chunks where the conclusion was that the smaller the segment of data, the more similarities existed between them. These findings might be used to make conclusions regarding the possibility of similarities between developers or similar security concerns on local and international websites. We could assume that the patterns of the vulnerabilities similarity between Bangladeshi and foreign websites are due to the same development practice or similar security vulnerabilities or even similar regional development factor by calculating these common patterns. This not only points out similarities but also helps in the development of a broader perception of web security that can give clues on how the websites of various regions can be interlinked.

4.2.4 Summary of Preprocessed Data

The research creates a complete dataset which contains data from both Bangladeshi and international websites to discover web application security weaknesses through machine learning methods. Website URLs are collected from two reliable sources, BASIS and Kaggle and the links were manually checked for a clean dataset. An automated security system operates by sending HTTP/HTTPS requests and it analyzes server responses to discover security vulnerabilities. The system created a first set of vulnerabilities which appeared as human-readable text and then transformed those vulnerabilities into binary formats that showed 1 for existing vulnerabilities and 0 for missing vulnerabilities. Other automated scripts were created to gather data about backend programming languages and frameworks and server types and website categories and geographical location.Manual checks were performed with Wappalyzer because some websites blocked access and prevented them from gathering complete metadata.Unidentified data were classified as unknown because technical issues prevented complete identification of the information.

One-hot encoding was performed to transform categorical metadata features into numerical format for machine learning model training. The vulnerabilities were categorized into six distinct categories which shared common attack methods. The approach enhanced model interpretation through dataset simplification and vulnerability grouping into six logical categories.
UMAP and t-SNE create visualizations which show how websites are connected through their geographical location and backend technologies and frameworks.Clustering patterns were displayed by this which showed how technology infrastructure and territorial trends have commonalities.

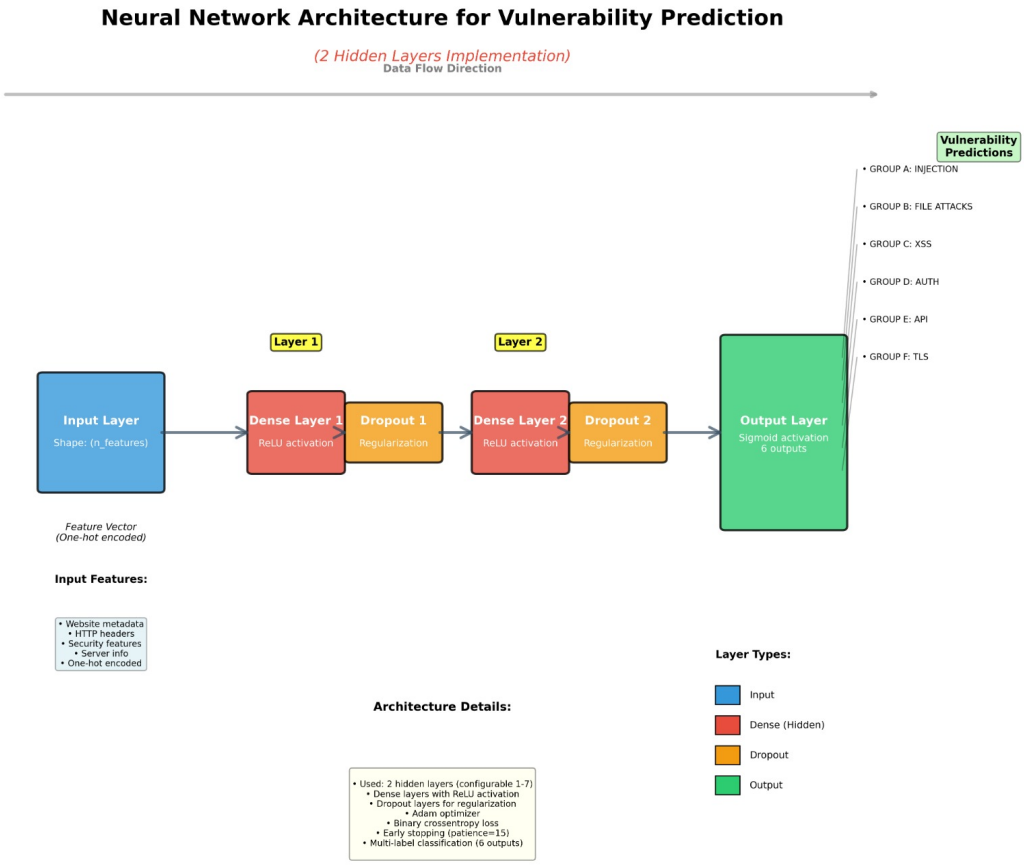

Figure 4.9: Vulnerability Prediction model Architecture

4.3 Implementation of Selected Design

4.3.1 Vulnerability Prediction model

Data Processing for Vulnerability Prediction model

The process begins by loading the raw dataset, which may include missing values. These missing values are replaced with zeros to ensure uniformity. This preprocessing stage is necessary to prevent errors during training due to missing data. The data is then standardized using StandardScaler to normalize the input features. This is a crucial step in the preprocessing pipeline because neural networks require features to be on a similar scale to learn effectively. When features vary greatly in magnitude, the network may overemphasize some features at the expense of others, which can slow convergence and hinder optimal learning.

Data Splitting for Vulnerability Prediction model

A train test split by scikit-learn is used to split the dataset with 20% of the data being used as testing and 80% being used as training data. The 80/20 ratio is a ratio that is widely applied in machine learning, it offers a balance between sufficient data to train the model and sufficient data to test the model. Why 80% and not 70% or 60%? It is a well-known fact that an 80/20 split is chosen because it gives enough data with which to train and makes the test set big enough to draw meaningful conclusions. The training with 70 percent or 60 percent would result in an inadequate amount of data to be used in training the model, thus, risking underfitting. Conversely, a larger testing data (e.g. 50% test) would leave the model with fewer examples to learn on, and this would impair the generalization ability of the model. This division is confirmed by science because it can be considered a decent trade-off between the adequacy of training data and the necessity of evaluation data.
In case of class imbalances in the datasets, stratified sampling will make sure that the training and the test sets are reflective of the general distribution of the target labels. This is of particular concern in the vulnerability detection tasks in which the dataset might have more non vulnerable sites compared to vulnerable ones. Stratification ensures that the model is exposed to a fair representation of vulnerabilities of both the training and the test set to avoid biased model predictions.

Model Architecture 1

This model consists of two hidden layers in its neural network, the first of which has a higher number of neurons (128) than the second (64), to find a balance necessary between overfitting and complexity. Using three or more hidden layers could cause the model to overfit if the size of the data is relatively small, therefore it is prudent to use only two layers in a given neural network model. While the two hidden layers can adequately describe the complex, non-linear relationships between the input features of the metadata set with the resulting vulnerabilities, this model will not be as difficult to implement as other designs with greater complexity.
The first hidden layer contains 128 neurons so that this number represents enough learning power in order to learn the various basic relationships between the 61 metadata feature inputs. With a lower number of neurons (for instance, 64), the network would be unable to learn all of the potentially complex relationships that

could exist in the input data and would lead to underfitting. Conversely, the second hidden layer's 64 neurons serves as a safeguard against overfitting by restricting how many higher-level features can be learned by the network from the input data.
After experimental evaluation of various architectural depths, the two-layer design is found to be an optimal solution. The architecture of the second layer performed better than the deeper models (the fourth layer achieved recall = 0.6673 and F1-score = 0.6481; the sixth layer achieved recall = 0.6434 and F1-score = 0.6300) with a recall of 0.7241 and an F1-score of 0.6872. The observed trend of performance decline with deeper models (the fourth and sixth layers) indicates that adding additional hidden layers does not inherently increase the predictive performance but adds additional complexity. This corresponds to the concept of diminishing returns on the depth of neural networks. The best hyperparameter configuration was shown to consist of two hidden layers with 128 and 64 neurons respectively. The result was an optimal trade-off between sufficient representational capacity (to learn the pattern of vulnerabilities) and sufficient generalization performance. The deeper models with more layers were overfitting, demonstrated by decreased recalls, an important metric for measuring the false negative rate associated with security vulnerability detection. These experiments validate the theoretical justification for limiting the depth of a neural network to two hidden layers, demonstrating that using this network architecture optimizes the model's ability to detect true vulnerabilities without reducing the accuracy of its predictions or incurring the excessive computational load associated with unnecessary depth in neural networks.
The application of ReLU (Rectified Linear Unit) as the activation function in the hidden layers can be justified by the fact that besides bringing non-linearity into the network, it is also computationally efficient. It is proved that ReLU helps to achieve quicker convergence than other activation functions such as sigmoid or tanh because it removes the vanishing gradient issue [5]. In addition, ReLU is computationally easier than tanh and sigmoid, and hence a beneficial deep neural network choice.

Dropout Regularization

One of the techniques used to prevent overfitting is dropout that randomly removes a fraction of the neurons during training. The dropout rates are set to 0.3 and 0.2 in the first and second hidden layers respectively in this model. The first hidden layer dropout is 0.3 to avoid the model overfitting to noisy and less abstract input features. A dropout rate of 0.3 was selected as it is sufficiently regularizing without the model losing the necessary information [8]. The excessive use of dropout (such as 0.5 or higher) would limit the ability of the model to fit complicated patterns, causing underfitting. On the other hand, the dropout applied would need to be too small (Say 0.1) and it would lead to the probability of overfitting.
A dropout rate of 0.2 follows the second layer. At this point, the model has already acquired more abstract features and it does not need much regularization. The reduced dropout rate of 0.2 is useful in preserving the generalization property of the model without impeding its capacity to acquire the higher level interactions using the data.

Output Layer: Sigmoid Activation In the case of the output layer, a sigmoid activation function is applied, which generates 6 neurons, each of which corresponds

to a vulnerability group. This is suitable in multi-label classification where the multiple outcomes indicate the likelihood of a particular vulnerability to be in a web application. The sigmoid activation is important in that it allows each output neuron to give a separate probability that is 0–1 which is good when there are multiple labels of vulnerability and they are not all mutually exclusive. Sigmoid activation also enables the model to address the vulnerabilities individually and estimate the probability of each vulnerability to be present or absent.[10]
The use of softmax would not be suitable in this case since the results are normally applied in multi-class classification where only a single class is being predicted and the results or outputs should add up to 1. Multi-label classification on the other hand involves independent prediction, with each output being a binary choice. ReLU is typically applied to hidden layers however not to the output layer in a classification task since it generates unlimited values contrary to sigmoid that has a limit of 0 to 1, perfect in probability model [6].

Understanding Binary Cross-Entropy and Its Application in Vulnerability Prediction

Binary Cross-Entropy (BCE) operates as a common loss function which binary classification tasks. The function provides a prediction which calculates the probability that an instance belongs to a specific class. Multi-label classification problems require researchers to use Binary Cross-Entropy because the method allows them to treat each label as a distinct binary classification task. This approach enables the model to make multiple predictions about different classes from each input.
Multiple vulnerabilities can exist at the same time on websites because they present multiple security weaknesses. A website can be attacked through Injection Attacks and Cross-Site Scripting (XSS) while other security weaknesses exist. The task requires multiple labels because a single website can have multiple vulnerabilities which need to be predicted through separate predictions.
Binary Cross-Entropy serves our needs because it enables the model to predict each vulnerability through independent binary classification tasks. The system used Categorical Cross-Entropy to handle multi-class classification because this method only permits a single label assignment for each sample.

Binary Cross-Entropy vs. Categorical Cross-Entropy for Multi-Label Vulnerability Prediction

Binary Cross-Entropy (BCE):

The application functions in multi-label classification because it treats each vulnerability label as an individual binary classification task.
The model uses Sigmoid activation function across all output nodes. The output generates independent probabilities that range from 0 to 1 enabling the model to forecast multiple vulnerabilities simultaneously.
The system computes BCE loss for every output label because it treats each output label as a separate entity without assuming exclusive output relationships. The requirement exists because websites can experience multiple vulnerability types at the same time which security experts must identify.

Categorical Cross-Entropy (CCE):
The application functions as a multi-class classification tool which assigns each sample to one specific class. The system permits only one label to be active at any given moment.
The output nodes use Softmax activation function which makes all class probabilities total 1.0 to provide their predictions. The model must predict only one class which is vulnerability because multiple labels can be correct in multi-label situations.
The Categorical Cross-Entropy loss function requires that only one class can be true but this requirement makes it inappropriate for situations where multiple classes (vulnerabilities) can be true at the same time.

Key Differences:

1. Problem Type:
The Binary Cross-Entropy loss function works for multi-label classification because a website sample can belong to more than one class.
The Categorical Cross-Entropy loss function operates correctly in multi-class classification situations because it requires samples to belong to only one class at any given time.

2. Activation Function:
The Binary Cross-Entropy function employs Sigmoid activation to produce independent prediction results for each of the binary classification tasks which treat vulnerabilities as separate identification challenges. The Categorical Cross-Entropy function uses Softmax activation to establish output probabilities which must total 1.0, thus restricting the model to predict only one vulnerability class at any moment.

3. Output Interpretation:
Binary Cross-Entropy: Each output node (vulnerability) produces a probability score between 0 and 1, indicating the likelihood of that specific vulnerability being present. The outputs are independent which allows the model to predict multiple vulnerabilities at the same time.
Categorical Cross-Entropy: The output is a single probability distribution across all classes which forces the model to select one class that has the highest probability but this approach does not work for multi-label tasks.

Example:

Binary Cross-Entropy (Correct for Multi-Label Classification):
The model reaches its decision about Website #1 based on actual labels [1, 0, 1, 0, 1, 0] because they are vulnerable to Group A through Group C and Group E attacks.
The model produces its output through Binary Cross-Entropy which generates these results:
Model output (Sigmoid): [0.85, 0.12, 0.78, 0.23, 0.91, 0.34]

The probabilities for Group A show an 85 percent probability which leads to a Yes (1) answer. The probability for Group C shows a 78 percent probability which leads to a Yes (1) answer. The Group E probability shows a 91 percent chance which leads to a Yes (1) answer. The groups B through D and F show lower probabilities which results in a No (0) answer. The system predicts threats through an output threshold set at 0.5 which shows [1 0 1 0 1 0] as its results.

Categorical Cross-Entropy (Incorrect for Multi-Label Classification):
The model using Categorical Cross-Entropy with Softmax activation shows these results for Website #1 which contains actual labels [1 0 1 0 1 0].
Model output (Softmax): [0.60, 0.05, 0.15, 0.05, 0.10, 0.05] (Sum = 1.0)
The model identifies Group A as the primary threat through its 60 percent probability assessment while it overlooks existing threats from Groups C and E. The output fails to provide accurate results because multiple vulnerabilities exist on the website yet Categorical Cross-Entropy restricts the model to present only one prediction.

Training Hyperparameters

The model is trained for 50 epochs with a batch size of 32 and a learning rate of 0.001. The choice of 50 epochs balances underfitting and overfitting because the model needs more training time which 50 epochs provide while the extra training time after 50 epochs causes the model to memorize training data instead of learning how to generalize. Empirical studies show that performance improvements diminish beyond this point, and overfitting may occur [10].
The system uses early stopping with a patience period of 15 epochs to prevent overfitting. Training stops if the validation loss does not improve for 15 consecutive epochs. This value is a compromise—too small a patience may stop training too early, while too large a value may allow overfitting.
The training process uses a batch size of 32 which creates an equal distribution between training speed and generalization ability. The use of smaller batch sizes creates a slower convergence speed which results in longer training times while the use of larger batch sizes creates lower model generalization because the model updates occur at less regular intervals. .[7]

Learning Rate:

The Adam optimizer is typically used with the learning rate of 0.001, because it allows the learning to be adaptive to work better on complex datasets, unlike the traditional methods that use constant learning rate [7]. The preference of Adam over other models due to its efficiency in processing big data and sophisticated construction models is because it is used in such tasks as cybersecurity vulnerability detection, which is the multi-label classification task.

Vulnerability Labels

The labels of vulnerabilities in this model are associated with certain types of web application vulnerabilities. These labels are binary signs (0 or 1) that indicate the status of a given vulnerability or the absence thereof to each web application.

Input

The neural network input consists of metadata characteristics of the dataset. These characteristics are some of the characteristics of the web applications like the server information, programming languages, security measures, and file formats. It can also have location-based characteristics, such as specifying whether the web application is local in Bangladesh or not. Predictions of vulnerability in web applications are done using these features and the StandardScaler is used to normalize the features such that all the features have equal influence to the model and none of the features reduces the other because of their varying scales.

Output

The neural network has 6 neurons as the output layer, which represent the vulnerability groups:

1. GROUP A: Injection & Input Attacks
2. GROUP B: File & Path Attacks
3. GROUP C: Cross-Site Attacks
4. GROUP D: Authentication & Access Control
5. GROUP E: API & Request Manipulation
6. GROUP F: ENCRYPTION & TLS ISSUES

4.3.2 Random Forest Model

Random Forest model is an ensemble learning method which uses the decision trees to perform classification and regression. It has been well known due to its strength and capacity to deal with both categorical and continuous data type. MultiOutputClassifier of scikit-learn is employed in this model to deal with the Multi labels characteristics of the vulnerability prediction task where each vulnerability group is separately predicted with a Random Forest classifier on each label.
Random Forest is most adequate with complex data as it can create a number of decision trees and average their predictions. This minimizes the variance in the model and it becomes less likely to overfit. [2]

Parameters

Number of Trees (n_estimators = 100):
The number of trees is adjusted to 100 to balance between the performance and the computational efficiency.

Max Depth (max_depth = 10):
The maximum depth of a tree is set at 10, which stops the possibility of overfitting since the trees will not represent some noise or overly complicated patterns in the training set .[2]

4.3.3 Gradient Boosting Model

Another ensemble method is the Gradient Boosting technique which is a sequential method of building decision trees, where each successive tree rectifies the errors of the predecessor. This method is very effective with structured/tabular data and has a high predictive power of the model [3] . GradientBoostingClassifier of the scikit-learn is the boosting tree trainer in this model to predict vulnerability. Gradient Boosting is selected because it aims at reducing bias through the correction of the error committed by the previous learners. The method works especially well with high dimensional data such as vulnerability prediction problems because it is able to find the intricate links and interactions among the input features and the target labels [3]. It is extremely popular due to its ability to predict and efficiency in organized information.

Parameters:

Estimators (n estimators=100) : The number of estimators will be fixed at 100 in order to trade training time and accuracy.

Max Depth (maxdepth=5) : This puts restrictions on the trees in that it should not be too many. Increasing the depth of the trees would lead to overfitting on the training data (Friedman, 2001). The shallow trees provide the most significant patterns and do not overfit especially when dealing with limited data sets [9].

Learning Rate (learning rate=0.1): A learning rate of 0.1 yields a good balance between model stability and speed of convergence. The learning rate of a lower value will take more estimators to converge whereas a higher rate will make the model overshoot optimal solutions [3].

4.3.4 SHAP Explanation of Neural Networks

SHAP (SHapley Additive Explanations) offers a model of explaining the predictions of machine learning models. In the case of Neural Networks, DeepExplainer of SHAP is employed to compute Shapley values in an efficient manner. The DeepExplainer has been designed to work with deep learning models and can deal with non-linear relationships that are complicated and inherent to neural networks.

The benefits of DeepExplainer on Neural Networks
Neural networks are very non-linear in their relationships and non-linear interaction patterns cannot be represented using traditional methods such as Linear Regression or Decision Trees. The layer-wise relevance propagation (LRP) is a technique that

DeepExplainer uses to efficiently propagate the feature importances over the layers of the network [13]. It is computationally low and best suited to deep architectures since it can process complex, non-linear transformations and make explanations of models which take the form of activation functions such as ReLU. DeepExplainer is especially important in the case of neural networks which are frequently deep and consume large amounts of resources to compute.

Shapley Values of Fair Attribution:
The major advantage of applying Shapley values in SHAP is that they possess the ability to offer equitable attribution of the prediction to each feature. Based on the theory of cooperative games, Shapley values justly assign the input of each feature to the model output. This is essential in the determination of the effect of each feature on the eventual prediction [1]. SHAP will enable us to have fair and non-biased allocation of features and will make the decisions made by the model transparent and understandable.

SHAP Model-Agnostic Nature:
SHAP values are model-agnostic in that they are applicable to any model of machine learning. The latter is an advantage of SHAP as it can be used to gain an insight into complex models such as Neural Networks that are usually opaque in their decision making process. The model-agnostic property enables comparing feature importances of the various models, including Neural Networks, Random Forests, and Gradient Boosting.

Neural Networks SHAP Interpretability:
SHAP interpretability is essential in practice where knowledge of factors contributing to vulnerability prediction can be used to do remediation in areas including cybersecurity. A prediction may have the most influential features, e.g., the type of server, programming language, or encryption scheme, which can be visually analyzed with SHAP. This contributes to the fact that SHAP is a priceless resource to any stakeholder by having to interpret and believe in the predictions of the model.

4.3.5 SHAP Visualizations

SHAP offers a number of visualization methods to examine feature contributions in models. These are summary plots, dependence plots, and force plots which are especially convenient to explain the influence of certain features on predictions.

4.3.6 Interpretability and Decision Making

The SHAP model used to explain the Neural Networks makes the model easier to interpret and as a result, non-experts can comprehend the factors behind the prediction. This is important in applications of cybersecurity where the stakeholders, who could be security analysts or auditors, should understand the rationale of a prediction of a model in order to act appropriately.

Rationale of Interpretability:
Neural Networks have been considered as black-box models because they are complex. Through SHAP we offer transparency in the decision-making process of the model and therefore communication and justification of decisions of the model becomes simpler. This is relevant to cybersecurity because it will help instill trust in the model, particularly when the model is to be applied in making high-stakes decisions like detecting vulnerabilities in web applications [11].
Neural Networks explanation architecture is a more detailed and interpretative model of predicting models based on the SHAP architecture. Using DeepExplainer, Shapley values, and other visualization, one can draw significant conclusions about the factors which drive predictions. This interpretability is vital to the use of complex models, especially in such areas of operational activity as cybersecurity, where transparency and trust cannot be ignored.

4.3.7 Neural Network Architecture for Dual Prediction Model

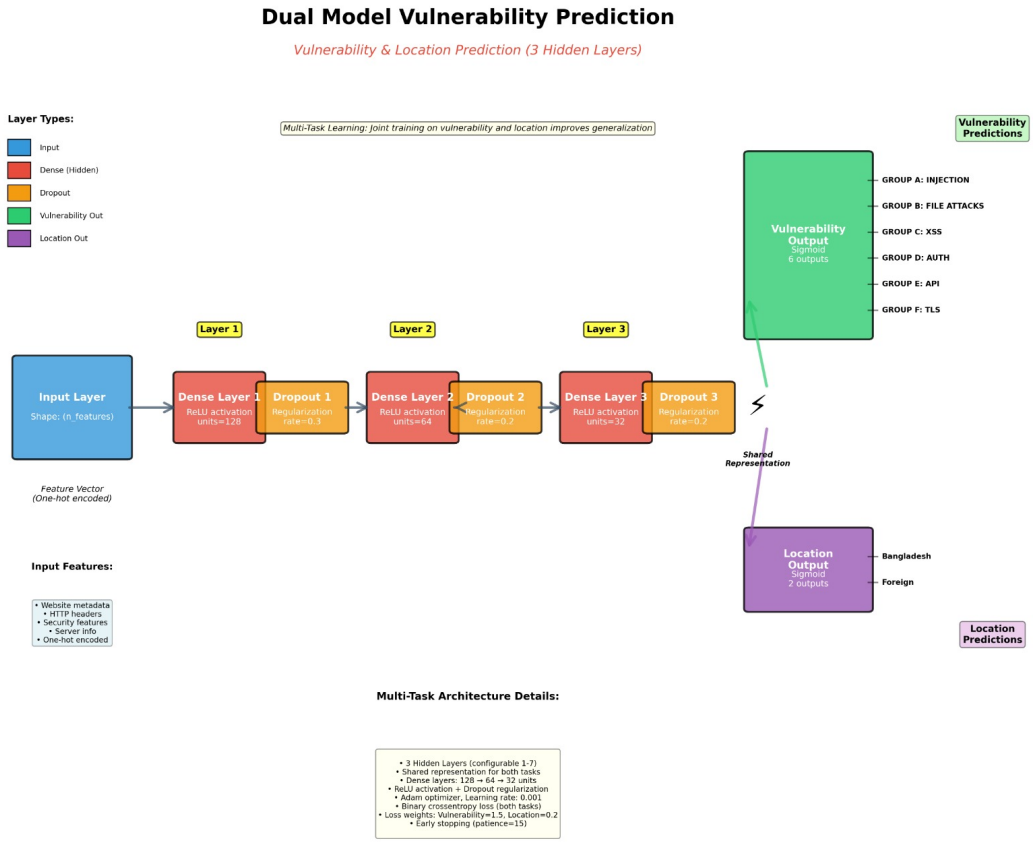

Figure 4.10: Vulnerability Prediction model Architecture

The neural network model used in this system operates as a multi-task learning framework which enables the model to simultaneously forecast web application vulnerabilities and their associated geographical locations in Bangladesh and other countries. The system architecture has been specifically developed to solve complex multi-output challenges through neural networks which analyze structured data

relationships.

Model Inputs and Outputs:

The model takes as input a feature matrix X which contains metadata information about each website. The metadata contains various attributes which include server type and programming language among others to predict both vulnerability and location results. The input shape requires metadata to specify all its available features.
The model has two distinct output layers. The Vulnerability Output generates a binary classification output that contains 6 neurons which each neuron predicts the probability of a specific vulnerability. The system uses a sigmoid activation function which generates output values that range between 0 and 1 to show the probability of each vulnerability existing in the system.

Location Output:

A binary classification output with 2 neurons, predicting whether a website originates from Bangladesh or from a foreign country. The system uses the sigmoid activation function to produce location category probabilities.

Model Architecture:

The model architecture consists of several key components. The input layer takes the metadata features as input which determines the input shape based on the number of features in the metadata. The model enables users to select between one and three hidden layers for its design. The model needs three hidden layers to maintain proper data complexity while stopping the process of overfitting.
The model architecture consists of several key components. The input layer takes metadata features as input which creates the input shape through the total number of features. The model enables users to select between one and seven hidden layers; however, empirical evaluation revealed that three hidden layers provide optimal balance between model complexity and generalization performance. The three-layer configuration achieved the highest F1score of 0.7148 and accuracy of 0.7577 which surpassed both 2layer architecture (F1$\bar{0}$.6973) and 5layer architecture (F1=0.6985) performance. The three-layer architecture (128→64→32 neurons) creates an optimal feature hierarchy that avoids the information bottleneck observed in deeper networks because all layers in the five-layer configuration maintain 32 neurons which limits data representation capabilities. According to Bengio et al. (2013) deep learning models develop better performance through hierarchical feature learning but they experience vanishing gradients and diminished information flow problems when their depth increases without matching width extensions. The three-layer configuration enables hierarchical feature extraction through its mixed neuron structure that begins with 128 neurons and ends with 32 neurons.

The three hidden layers maintain proper data complexity while preventing overfitting. The system learns dangerous patterns by using three different levels of abstraction which start from the first layer that contains 128 neurons to the second layer which processes data through 64 neurons to the third layer that produces high-level vulnerability indicators with 32 neurons. The first hidden layer uses the ReLU activation function because it provides non-linear characteristics enabling faster convergence and requires fewer computational resources [10] . ReLU activation function prevents the vanishing gradient issue which affects networks with more than five layers. The three-layer system maintains consistent gradient movement while the five-layer system shows performance decline.

The system implements dropout layers with rates of dropout1=0.3, dropout2=0.2, and dropout3=0.2 to minimize overfitting. The decreasing dropout pattern (0.3→0.2→0.2) reflects that earlier layers benefit from more aggressive regularization as they learn general features, while deeper layers require less dropout to preserve learned abstract representations [8] . The five-layer setup showed excessive depth with equal dropout rates which led to over-regularization and underfitting according to its 3.76% recall decline (0.7214→0.6838) when compared to the three-layer model. The new neuron count in the deeper layers (128→64→32) creates abstract features which lead to improved generalization. The encoder architecture design patterns create an information funnel which forces the network to learn more discriminative features at every layer according to Hinton and Salakhutdinov 2006. The five-layer system used a bottleneck design which compressed information at an excessive rate during the first three layers and caused a loss of essential representational capacity in the last two layers.

The three-layer architecture achieved better recall results through its 0.7214 score than the five-layer system which achieved 0.6838 score because the balanced dropout method successfully stopped overfitting while maintaining its ability to detect vulnerabilities. The system achieves regularization through its standard dropout rates which maintain model performance according to established practices. The three-layer configuration required a learning rate of 0.001 and a batch size of 32 because this combination together with 15 epoch early stopping was the optimal training strategy which resulted in enough time for the system to achieve convergence while protecting against overfitting. The Adam optimizer's adaptive learning rate mechanism [7] works effectively with the three-layer architecture's gradient landscape whereas the five-layer model needs extended training periods or modified learning rate schedules to achieve similar convergence results which their performance regression demonstrates.

4.3.8 Loss Function and Metrics

The model applies binary cross-entropy as its loss function to both vulnerability and location outputs. The task requires binary classification for the presence or absence of vulnerabilities and the location classification which makes this loss function suitable.
The model uses binary accuracy as the evaluation metric for both vulnerability and location outputs. The Adam optimizer serves as the optimization method for the

model because it utilizes adaptive learning rates [7].

4.3.9 Epochs and Early Stopping

The model requires 50 epochs for its training process. The number of epochs is chosen based on empirical observations that 50 epochs provide a good balance between allowing the model to learn effectively and preventing overfitting.
The implementation of early stopping prevents the model from overfitting to the training data. The training process stops when the validation loss shows no further improvement for a designated number of consecutive epochs (patience = 15).

4.3.10 Evaluation

The model uses standard classification metrics for its evaluation which include accuracy, precision, recall and F1-score. The system calculates these metrics for each vulnerability group and the location prediction system. The results are then compared to the performance of other models, such as Random Forest and Gradient Boosting, to assess their relative effectiveness in predicting vulnerabilities and locations.

4.3.11 Random Forest and Gradient Boosting Models

The neural network model has strong capabilities for discovering complex non-linear patterns yet Random Forest and Gradient Boosting models serve as additional methods for evaluation. Random Forest creates an ensemble of decision trees which each tree learns from a unique data subset [2]. The method uses a different approach than Random Forest because it builds trees in a sequence which enables new trees to fix previous trees' mistakes [3].

4.3.12 SHAP for Model Explanation

SHAP provides model interpretation through SHapley Additive exPlanations which shows how different features combine to produce the model's output thus enabling users to understand how the model predicts vulnerabilities and their respective locations. TreeExplainer is used for Random Forest and Gradient Boosting models, while DeepExplainer is used for Neural Networks [13].

Chapter 5

Result Analysis

5.1 Performance Evaluation

This section presents a complete exploration of three machine learning models which include Neural Network Random Forest and Gradient Boosting used in the multi-label web application vulnerability detection assignment. The models were created and evaluated using a data set that contains six distinct vulnerability types which include injection and input attacks file and path attacks cross-site attacks authentication and access control vulnerability API and request manipulation vulnerability and encryption and TLS vulnerability.

Precision:
Precision is a measure of how many of the predicted vulnerabilities (or categories) are actually relevant. In the context of this dataset, if the model predicts a certain vulnerability group for a website, precision tells us how many of those predictions were correct. For example, if the model predicts a website is vulnerable to a specific vulnerability group, precision indicates what percentage of those predicted vulnerabilities are true positives. If the model predicts a vulnerability incorrectly (false positive), it negatively impacts the precision score. Precision is particularly important when the cost of false positives is high. In cybersecurity, a high precision score means that when a vulnerability is flagged, it is more likely to be a real issue, which can guide targeted mitigation efforts.

Recall:
Recall measures how many of the actual vulnerabilities (or categories) were correctly identified by the model. It focuses on the model's ability to identify all the relevant vulnerabilities in the dataset. In the context of vulnerability prediction, recall is crucial because missing a vulnerability (false negative) could have serious consequences. A high recall score means that the model is good at identifying most of the actual vulnerabilities, minimizing the chances of overlooking potentially dangerous vulnerabilities. However, a model with high recall might sacrifice precision, leading to more false positives (where the model flags vulnerabilities that aren't actually present).

F1-Score:
The F1-score is the harmonic mean of precision and recall, balancing the trade-off

between the two. In your case, an F1 score reflects the overall ability of the model to correctly predict vulnerabilities while minimizing both false positives and false negatives. This metric is especially useful when there is a class imbalance, which is often the case in vulnerability prediction tasks. For example, vulnerabilities like SQL injection may be much less frequent in a dataset compared to more common vulnerabilities. An F1 score that is balanced ensures that the model does not ignore the less frequent vulnerabilities while still performing well on the more common ones.

Accuracy:
In this models per element accuracy is being used to measure accuracy of the models. Per-Element Accuracy functions as an essential measurement tool that assesses how well a model performs in multi-label classification tasks which include predicting vulnerabilities. The traditional accuracy metric assesses a model's ability to classify samples correctly across the entire dataset, whereas Per-Element Accuracy evaluates the accuracy of each label prediction across multiple samples. The metric applies to the vulnerability prediction problem because every website sample in this task can receive multiple vulnerability labels at the same time.

Per-Element Accuracy Example:
The dataset which contains five test samples shows six different vulnerability groups present in each sample. The ground truth labels and the model's predictions are provided below:

Ground Truth Labels (y_true):

```
[1, 0, 1, 0, 1, 0],   Sample 1
[0, 1, 0, 1, 0, 1],   Sample 2
[1, 1, 1, 1, 1, 1],   Sample 3
[0, 0, 0, 0, 0, 0],   Sample 4
[1, 0, 1, 0, 0, 0],   Sample 5
```

Model Predictions (y_pred):

```
[1, 0, 1, 0, 1, 0],   Sample 1: All correct
[0, 1, 0, 0, 0, 1],   Sample 2: 5/6 correct
[1, 1, 1, 1, 1, 1],   Sample 3: All correct
[0, 0, 0, 0, 0, 1],   Sample 4: 5/6 correct
[1, 0, 1, 0, 0, 0],   Sample 5: 5/6 correct
```

The calculation of Per-Element Accuracy will be conducted as follows:
Sample 1 has all 6 labels correct, contributing 6 correct predictions.
Sample 2 has 5 out of 6 labels correct, contributing 5 correct predictions.
Sample 3 has all 6 labels correct, contributing 6 correct predictions.
Sample 4 has 5 out of 6 labels correct, contributing 5 correct predictions.
Sample 5 has 5 out of 6 labels correct, contributing 5 correct predictions.

Total Correct Predictions $= 6 + 5 + 6 + 5 + 5 = 27$
Total Predictions $= 5\,\text{samples} \times 6\,\text{vulnerabilities} = 30$

$$\text{Per-Element Accuracy} = \frac{27}{30} = 0.90\,\text{or}\,90\%$$

The model achieved correct predictions for 90 percent of the labels throughout all samples according to this information.

Per-Element Accuracy is Critical for Vulnerability Prediction:
The Per-Element Accuracy metric delivered superior model performance measurement results for vulnerability prediction tasks compared to standard accuracy metrics. The Per-Element Accuracy metric enables testing of every vulnerability category through its ability to assess each group of vulnerabilities. The model's detection capacity for various vulnerabilities serves as the primary factor that determines its overall testing success.
Multiple Label Management: The multi-label classification system allows a sample to belong to multiple categories simultaneously. Per-Element Accuracy enables the model to be evaluated for its individual prediction for each label (vulnerability group), even if the sample has multiple true labels. The system can better estimate how well it detects multiple vulnerabilities that appear together on one website.
Per-Element Accuracy enables assessment of model performance for every vulnerability type through its requirement to measure model outcomes on each specific vulnerability. The model uses the actual data distribution to develop its predictions for upcoming events.

5.2 Analysis of Design Solutions

The table in 4.1 displays the test results which show how each of the three models performed on test data. The assessment employed four standard classification metrics which included accuracy and precision and recall and F1-score to assess all six vulnerability groups while taking into account the potential class imbalance found in the multi-label classification challenge.

	Model	Avg Accuracy	Avg Precision	Avg Recall	Avg F1-Score
0	Random Forest	73.6%	56.1%	67.3%	0.599
1	Gradient Boosting	71.9%	53.3%	61.3%	0.557
2	Neural Network	72.1%	54.7%	63.8%	0.566

Figure 5.1: Overall methodology overview of the proposed system

Understanding Performance Metrics in the Context of Multi-Label Vulnerability Detection

1. Random Forest

The Random Forest model achieves its highest accuracy level through 73.6% accuracy measurement. The model delivers the most accurate performance between "Safe" and "Vulnerable" category predictions among the three tested models.

The Random Forest model achieves 56.1% precision which stands as the second best performance among all models because it successfully identifies positive cases (specifically "Vulnerable") 56.1% of the time. The result appears acceptable for this model yet the system needs further development to reduce its occurrence of false positive outcomes.

The model achieved an average recall of 67.3% which is the highest among the three models. The Random Forest model successfully detects "Vulnerable" cases but it still fails to identify 32.7% of actual cases.

Random Forest achieves its average F1-Score of 0.599 because it manages to maintain a balance between precision and recall. The F1-Score shows the model achieves a balanced performance between the two metrics yet the system needs better precision results.

2. Gradient Boosting

The average accuracy of Gradient Boosting reaches 71.9% which brings him slightly below Random Forest because he shows reduced capacity to classify both groups of data. The three models show precision results by which Gradient Boosting reaches his lowest performance at 53.3% precision. The model shows an increased tendency to make false positive errors because it predicts many "Vulnerable" cases when the actual status is "Safe."

The model achieves an average recall of 61.3% which is the second-highest among the three models. The model enables Gradient Boosting to identify 61.3% of actual "Vulnerable" cases while it fails to detect 38.7% of those cases.
The F1-Score for Gradient Boosting shows 0.557 as his lowest value among the three models. The model shows less favorable performance because it achieves lower F1-Score results which lead to better recall results but worse precision outcomes when compared with other models.

3. Neural Network
The Neural Network model shows an average accuracy of 72.3% which results in a middle performance level because it outperforms Gradient Boosting yet falls short of Random Forest. This shows that the Neural Network system achieves decent accuracy but does not deliver trustworthy results for this particular situation.
The Neural Network model achieves an average precision of 54.5% which enables it to exceed the performance of Gradient Boosting but fall short of Random Forest. The model shows low reliability for predicting the "Vulnerable" class because it continues to make numerous misclassifications.
The average recall of 65.9% puts Neural Network in third place because it can detect actual "Vulnerable" cases. The system shows better performance than Gradient Boosting because it detects more cases, yet it still fails to identify 34.1% of "Vulnerable" cases which creates space for better performance.
The Neural Network model shows an average F1-Score of 0.572 which makes it less effective than Random Forest but more effective than Gradient Boosting. The model provides adequate harmony between precision and recall yet needs extra optimization work to enhance the connection between both metrics.

5.3 Final Design Adjustments

Using different Layers

Layer 2 Configuration:
The system contains Layer 2 design settings which establish the following performance metrics. The system achieves an accuracy rate of 72.3 percent. The system shows a precision rate of 55.3 percent. The system achieves a recall rate of 68.2 percent. The system produced an F1-Score result of 0.582.

Layer 4 Configuration:
The system achieved 72.3 percent accuracy. The system reached 55.3 percent precision accuracy. It obtained 65.9 percent recall performance. The F1-Score evaluation resulted in a score of 0.572.

Layer 6 Configuration:
The system achieved an accuracy rate of 72.2 percent. It attained a precision rate of 54.6 percent, achieved a recall rate of 69.4 percent. Also produced an F1 score of 0.574.

Analysis of Results:
Accuracy: Both Layer 2 and Layer 4 achieve the highest accuracy at 72.3% while Layer 6 falls slightly behind at 72.2%. The comparison results remain mostly unchanged because of the minimal difference in accuracy between the two samples.
Precision: Layer 2 and Layer 4 both achieve a precision value of 55.3% which leads to their better performance than Layer 6 which has a 54.6% precision value. The model needs to achieve higher precision because it must prevent all false positive errors which become critical when handling multi-label tasks that involve identifying non-existent vulnerabilities.
Recall: Layer 6 has the highest recall at 69.4%, followed by Layer 2 at 68.2%, and Layer 4 at 65.9%. Recall measures the ability of the model to identify all true vulnerabilities, and while Layer 6 excels here, the performance difference between Layer 2 and Layer 6 is quite small.
F1-Score: The F1-score reaches its highest value at 0.582 with Layer 2 while Layer 6 follows at 0.574 and Layer 4 reaches 0.572. The F1-score measures the balance between precision and recall, and since Layer 2 attained the highest F1-score, it demonstrates superior capabilities for detecting vulnerabilities without producing false positive results.
Layer 6 shows higher recall results at 69.4% while Layer 2 shows lower results at 68.2% which demonstrates that Layer 2 has better precision results at 55.3% than Layer 6 which achieved 54.6% and better F1-score results at 0.582 than Layer 6 which reached 0.574. The best configuration for this multi-label vulnerability prediction task exists in Layer 2 because it provides the ideal combination of precision and recall with F1-score. While Layer 6 excels at detecting more test cases, Layer 2 delivers superior overall performance because it detects vulnerabilities effectively while reducing false positive rates through its balanced design.

5.4 Statistical Analysis

SHAP Explainability Analysis for Vulnerability Prediction
The research team conducted SHAP analysis on six different vulnerability groups to explain model predictions and identify which features had the greatest impact on prediction results. SHAP values quantify the contribution of each feature to the model's predictions based on cooperative game theory principles, providing transparent insights into the decision-making process. The analysis identifies both technical factors and contextual elements that determine how different attack types are classified as vulnerabilities.

GROUP A: Injection & Input Attacks

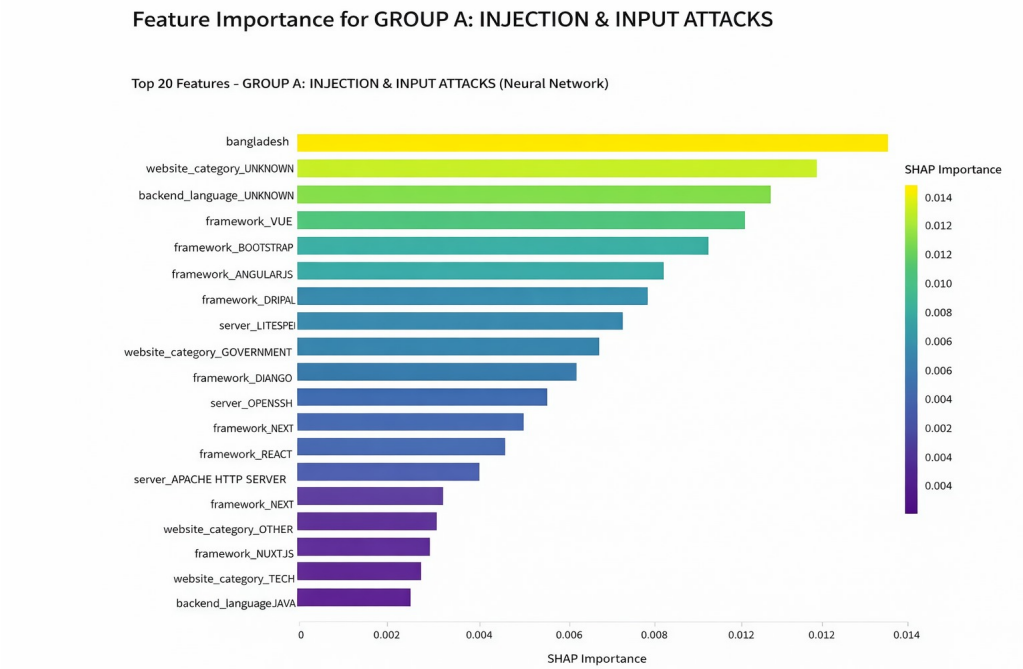

Figure 5.2: GROUP A: Injection & Input Attacks

Figure presents the SHAP feature importance analysis for injection and input attack vulnerabilities, including SQL injection, command injection, and related input manipulation attacks. In terms of injection and input attacks, the maximum SHAP value for geographic features is attributed to Bangladesh with a value of 0.016, meaning sites based out of Bangladesh have been found to be more prone to injecting attacks. The second highest SHAP value comes from 'website_category_UNKNOWN' at 0.0145; this suggests the danger of having poor or no documentation about a website and having no security standards during the web development process. Finally, Java as a backend programming language has been determined to be the least likely to provide protection against injection-type vulnerabilities with a SHAP score of 0.002; this indicates that the actual coding style and/or framework used in creating the codebase will have a larger impact than the selection of the programming language when attempting to mitigate injection attacks.

GROUP B: File & Path Attacks:

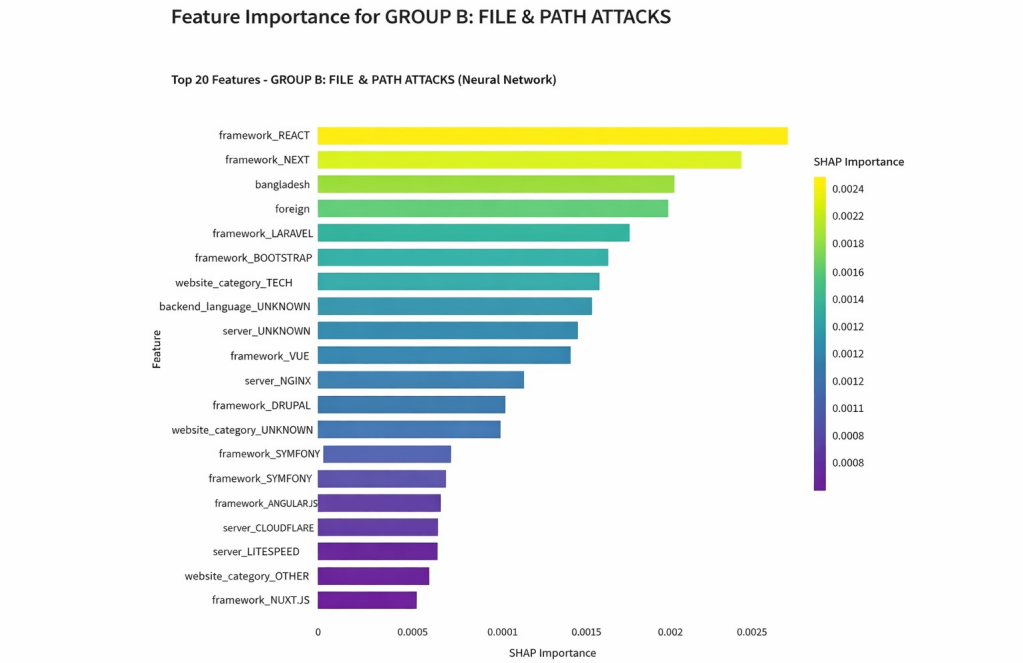

Figure 5.3: GROUP B: File & Path Attacks

Figure illustrates the SHAP feature importance for file and path attack vulnerabilities, including directory traversal, local file inclusion, and arbitrary file upload vulnerabilities. In this study, we found that modern front-end frameworks, such as React and Next.js, were important contributors to File/Path vulnerabilities (e.g., React (0.0024), Next.js (0.0023), these vulnerabilities stem from the dynamic nature of clientside routing and server-side rendering, as well as the inherent risk associated with dynamically generating file / path references; while back-end languages like Java demonstrate minimal significance (0.001), therefore reinforcing the idea that File Handling Security is primarily a framework architectural concern and not necessarily dependent on back-end language choice.

GROUP C: Cross-Site Attacks:

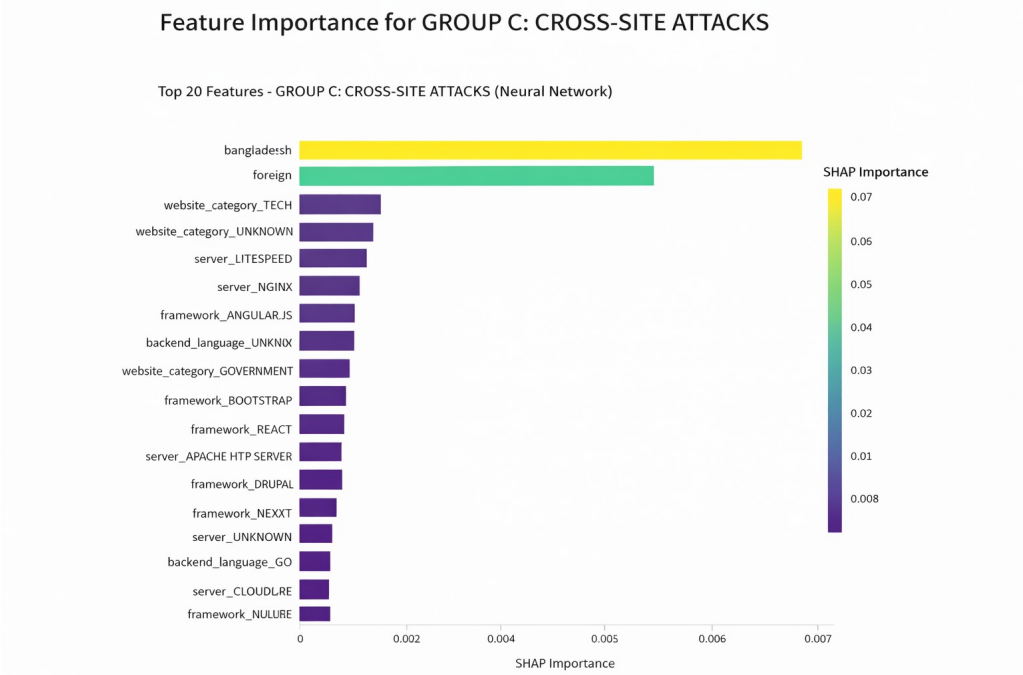

Figure 5.4: GROUP C: Cross-Site Attacks

Figure presents the SHAP feature importance for cross-site scripting (XSS) and cross-site request forgery (CSRF) vulnerabilities. Cross-Site (XSS) and Cross-Site Request Forgery (CSRF) vulnerabilities have been shown to have a noteworthy geographical influence on organisations. The largest feature ranking related to XSS and CSRF risk was awarded to Bangladesh (0.07), establishing a clear local bias between countries regarding the output encoding and sanitisation process. Conversely, the use of the React framework for developing web applications was rated low within the feature importance vector (0.007), implying that geographical location and development procedures influence the use of a framework more than its inherent level of protection against XSS attacks.

GROUP D: Authentication & Access Control

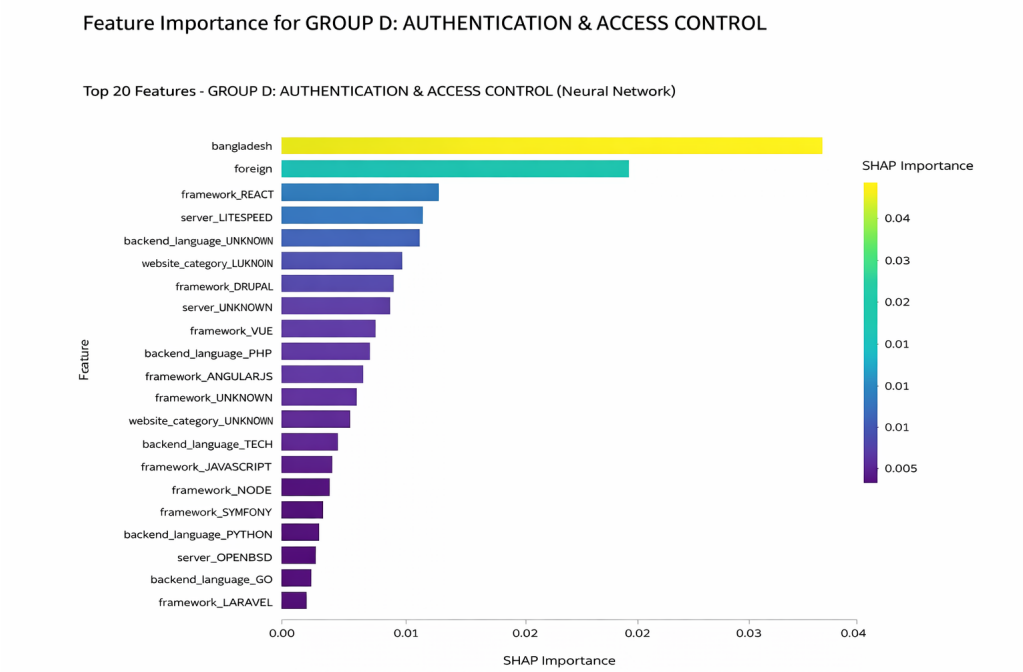

Figure 5.5: GROUP D: Authentication & Access Control

Figure displays the SHAP feature importance for authentication and access control vulnerabilities, including broken authentication, session management flaws, and privilege Similar to authentication vulnerabilities, geographic regions are factors contributing to the overall influence on AA or authorization vulnerabilities, with the Bangladesh feature (0.04) being one of the significant predictors. The existence of global standards such as OAuth 2.0 and JWT has minimised the regional variations in AA or authorization practices. The low SHAP value (0.008) of the Ruby on Rails framework indicates that the implementation of security features used for Authentication and Authorization is more closely related to the implementation process than the specific framework used.

GROUP E: API & Request Manipulation:

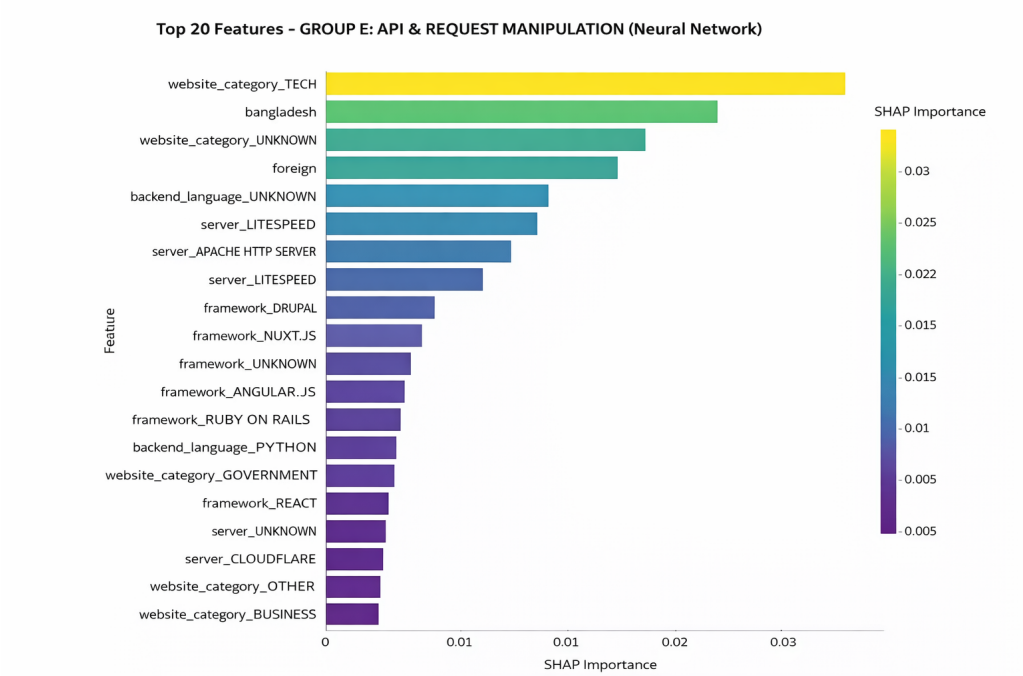

Figure 5.6: GROUP E: API & Request Manipulation

Figure illustrates the SHAP feature importance for API and request manipulation vulnerabilities, including serverside request forgery (SSRF), XML external entity injection, and insecure deserialization.
The website category "TECH" was determined to have the strongest correlation with API Vulnerability, with a score of (0.035). This result illustrates how complexly structured and integrated into third-party systems most technology-related websites are, resulting in greater risk of API Misuse. In contrast, the geographical area "Bangladesh" was shown to have a much lower effect on API Vulnerability than TECH at (0.027). This finding demonstrates how API Security is an international/global issue.

GROUP F: Encryption & TLS Issues:

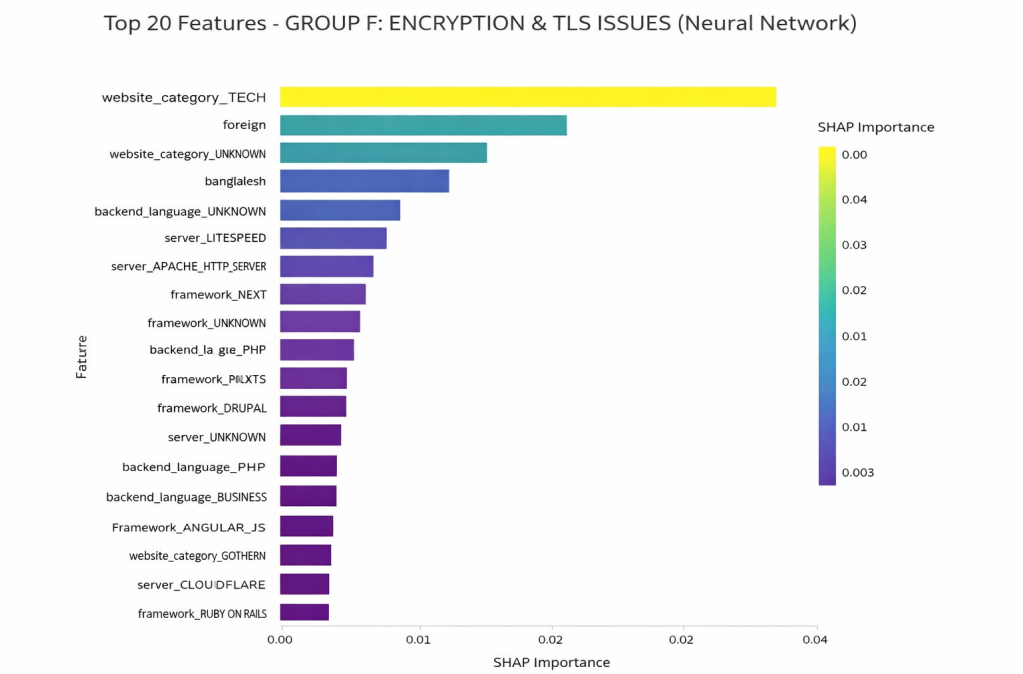

Figure 5.7: GROUP F: Encryption & TLS Issues

Figure presents the SHAP feature importance for encryption and transport layer security vulnerabilities, including weak cipher suites, certificate validation issues, and missing HSTS
The encryption vulnerabilities were mainly domain driven, with the weakest predictor as "website category – tech" (0.045). This shows significant Encryption vulnerabilities exist among tech companies because of their complexity in many areas including Third party Integrations and Infrastructure, even though these companies have a wealth of technology based expertise to aid in protecting them from this risk. The framework for Ruby on Rails, showed almost negligible effect (0.007) when predicting Encryption Vulnerability, supporting the concept that TLS is more reliant on infrastructure and Organisational Policies than on Framework Selection.

Cross-Group Synthesis and Strategic Implications

From the SHAP analysis, there are three main themes:

1. **Geographic Disparities:** Injection and authentication vulnerabilities demonstrate regional differences across Bangladesh where developer training and security practices differ significantly.

2. **Domain-Driven Vulnerabilities:** API and encryption vulnerabilities are correlated more to complexity of applications and Internet time than region.

3. **Lack of Documentation:** Lack of documentation pertaining to technology stack ("website_category_UNKNOWN") is associated with a higher likelihood of having a vulnerability, which underscores the importance of having a governance structure for security documentation.

In summary, the SHAP analysis provides an interpretation of how the Neural Network is structured as a Decision Support System (DSS) that will ultimately assist with the distribution of resources that are secure by design and support the allocation of those resources on a basis of specific vulnerability types and the context of each organization. on a basis of specific vulnerability types and the context of each organization.

Confusion Matrix:

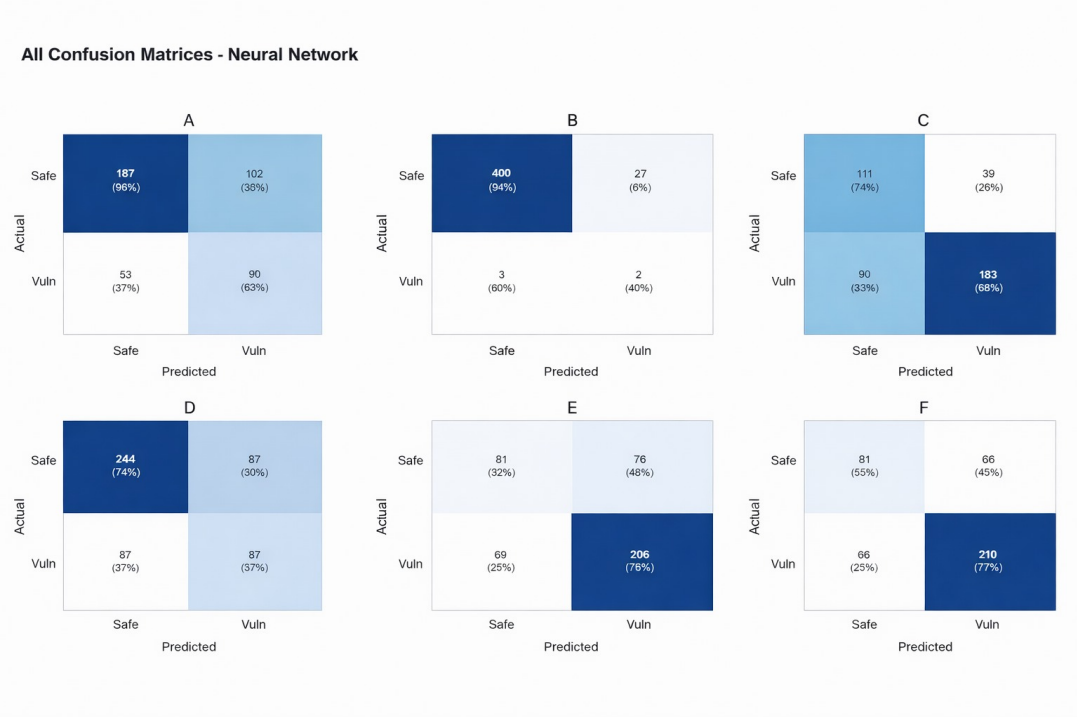

Figure 5.8: Vulnerability Prediction model confusion matrix

Group A: Safe Predictions: The model achieved a 64% success rate when it identified 186 Safe cases from Group A while 103 Safe cases of those 289 cases were incorrectly marked as Vulnerable. The results show that the system makes moderate errors when identifying Safe cases because it wrongly identifies almost one third of all Safe cases tested.
Vulnerable Predictions: The system incorrectly classified 55 of 143 actual Vulnerable cases as Safe which resulted in a 38% error rate while it successfully identified 88 instances as Vulnerable which made up 62% of the cases. The model demonstrates a major problem because it frequently misidentifies Vulnerable cases as Safe which results in decreased precision for the Vulnerable category.

Group B: Safe Predictions: The model achieved outstanding performance in Group B because it accurately identified 401 "Safe" cases out of 427 actual cases which resulted in a prediction accuracy of 94% and it misidentified 26 cases as "Vulnerable" which accounted for 6% of the total.
Vulnerable Predictions: The "Vulnerable" instances showed 3 out of 5 proper predictions which identified them as "Safe" (60%) and 2 instances were wrongly identified as "Vulnerable" (40%). The model displays exceptional accuracy for "Safe" category predictions but it makes minor errors when predicting "Vulnerable" cases which shows slight misclassification.

Group C: The model accurately predicted 112 of 150 actual "Safe" cases as "Safe" so its success rate reached 75 percent yet 38 "Safe" cases were wrongly identified as

"Vulnerable." The model demonstrates acceptable performance when making "Safe" predictions but shows a significant error rate that affects the "Safe" category.
The model achieved 66 percent accuracy by predicting 185 out of 282 actual "Vulnerable" cases as "Vulnerable" while it wrongly classified 97 cases as "Safe." The model demonstrates better performance for "Vulnerable" cases yet it still makes incorrect predictions by classifying many cases as "Safe."

Group D: The model made correct predictions about 219 "Safe" instances which matched 331 actual "Safe" cases. The model shows difficulty making "Safe" predictions because it incorrectly identifies most of those cases.
The 31 "Vulnerable" instances were misclassified as "Safe" because 31 out of 102 actual "Vulnerable" cases were incorrectly identified. The model shows difficulty detecting vulnerabilities because it misclassifies "Vulnerable" cases as "Safe."

Group E: The model identified 87 of 157 actual "Safe" cases correctly while it misclassified 70 cases as "Vulnerable." The system demonstrates its inability to correctly identify "Safe" cases because it misidentifies multiple "Safe" cases.
The model accurately predicted 218 of 278 "Vulnerable" cases which resulted in a 79 percent success rate while 57 cases were incorrectly identified as "Safe." The system produces effective results for "Vulnerable" cases but it struggles with identifying "Safe" cases accurately.

Group F: Safe Predictions: The model achieved 53% accuracy when predicting 78 of 147 actual "Safe" cases in Group F. The system identified 69 cases as "Vulnerable," which accounted for 47% of the total predictions. The system has difficulties making "Safe" predictions because it wrongly identifies almost half of them as "Vulnerable" results.
Vulnerable Predictions: The system achieved an 80% accuracy rate by correctly identifying 227 of 285 actual "Vulnerable" instances as "Vulnerable" while 58 instances were incorrectly identified as "Safe." The system demonstrates high accuracy when handling "Vulnerable" cases yet still manages to wrongly identify many "Safe" cases.

5.4.1 Performance Dual Prediction Model

Performance Metrics of Dual Model Prediction

Model	Precision	Recall	F1 Score	Accuracy
Neural Network	0.708371 ★	0.721357	0.714805 ★	0.757716 ★
Random Forest	0.695652	0.703941	0.699772	0.745756
Gradient Boosting	0.700976	0.724106 ★	0.712353	0.753858

Figure 5.9: Dual Model Overall Performance

Dual Model Vulnerability Prediction Performance Dual Model Vulnerability Prediction Performance Table shows how effective it was to perform both vulnerability and location prediction at the same time within one architecture. The Neural Network achieved a precision rate of 70.84%, meaning that for example, 71 out of every 100 predictions for vulnerabilities turned out to be correct. This body of work has been extremely valuable from an operational security perspective in terms of the potential for false positives resulting in wasted resources and alert fatigue on the part of the security teams involved. The Neural Network further outperformed the other models across multiple metrics, achieving the highest recall (72.14%), F1 Score (71.48%), and accuracy (75.77%). This indicates that the Neural Network has the best ability to have a strong trade-off between sensitivity and specificity when detecting threats.

The Gradient Boosting demonstrated a very strong performance as well, achieving a recall of 72.41%, which was very similar to the Neural Network's ability to correctly identify true vulnerabilities. However, due to its lower precision rate (70.10%) and F1 Score (71.24%), there was a higher number of false positives created compared to the Neural Network. The Random Forest model exhibited good performance as well (69.57% precision / 70.39% recall); however, it produced the lowest F1 Score and accuracy (69.98% F1 / 74.58% accuracy).

Ultimately, the Neural Network's consistently high performance relative to all three metrics (precision, F1 Score, and accuracy) further substantiates its position as the leading model for dual-task prediction, leveraging the joint learning of vulnerability and location features in order to provide improved overall detection capabilities.

Performance Interpretation and Analysis

The results of the dual prediction model illustrate some clear patterns; however, the relatively small range of F1 scores (69.98%–71.48% and a small spread of 1.5%) indicates that all of the models capture similar underlying patterns of vulnerability

in the feature space. Thus, the similarities in the performance of the three models indicate a very strong reason to believe that the dual-task learning framework of combining both vulnerability and location prediction provides a robust baseline that benefits all machine learning models equally.

The superior performance of the Neural Network Model on all metrics, particularly on the Precision (70.84%) and Recall (72.14%), demonstrate the value of multi-task learning in this domain. Since the Neural Network Model was developed by collectively training on both the vulnerability detection and the location prediction, it developed much more general feature representations and improved both specificity and sensitivity. As a result, the joint optimization resulted in the model developing complementary patterns; thus, vulnerabilities can have characteristics dependent on the location, and features in the location domain can correlate to specific vulnerabilities.

The accuracy values of 74.58%–75.77% indicate the difficulty of predicting both a vulnerability's type and location at the same time as this dual-task prediction is much more difficult than just predicting the value of one of them separately. A neural network's ability to correctly predict both the type of vulnerability and where it is located for around 75% of test cases is very good considering how combinatorially difficult it is to predict the types of multiple vulnerabilities at the same time and also where a website is located.

The gradient boosting algorithm has been able to achieve a relatively good recall (72.41%), and good performance from the neural network shows how effective ensemble approaches can be in discovering true vulnerabilities when using the dual-task approach. The fact the neural network has a higher precision than gradient boosting indicates that its architecture is designed to minimize the number of false positives while simultaneously achieving a good sensitivity level, which is important for the reality of practical security applications in order to reduce either the operation costs arising from missed threats or the operational costs arising from false alarms.

Location Prediction Performance

Geographic Classification Results The table presents the location prediction metrics that compare the classification of Bangladesh versus foreign origin across all three models.

Location Prediction Comparison

Model	Precision	Recall	F1 Score	Accuracy
Neural Network	0.771886	0.775463	0.773672	0.773148
Random Forest	0.812183	0.727273	0.767386	0.775463
Gradient Boosting	0.858252	0.831818	0.727749	0.790509

Figure 5.10: Location Prediction Comparison

The results suggest a significant performance differential between location prediction (F1 scores from 72.77%-77.37%) and vulnerability prediction (F1 scores from 69.90%-72.41%). This difference is largely attributed to the binary classification of location (i.e., Bangladesh vs foreign), whereas the vulnerability task also includes 6 additional dimensions of labels resulting in a 6-D Multi-Label Classification Task. In addition, SHAP analysis has provided evidence for the presence of strong geographic signal within the feature space of the specific multi-label vulnerability classification task.

Model-Specific Location Prediction Characteristics

Compared to all other tested algorithms for location prediction, the Neural Network exhibited the best recall at 77.54% and F1-score at 77.37%. This indicates that the Neural Network is capable of producing geographic classifications in a balanced and effective manner. The multi-task architecture in the Neural Network appears to be especially well suited for location prediction because it is able to leverage the location correlations provided by the SHAP analysis, which identified a number of vulnerability features that were related to geographic locations (e.g., cross-site attacks and authentication vulnerabilities). In addition, the ability of the Neural Network to utilize the same representation for vulnerability patterns and geographic origin creates a synergistic learning that increases the ability of the Neural Network to predict location compared to the capabilities of single-task models.

Random Forest exhibited the second-highest precision for geographic classification at 81.21% and therefore produced the least number of false positive geographic classifications when compared to the Neural Network. However, this increased precision signifies a reduction in recall (72.73%) due to the fact that Random Forest used a more conservative geographic classification threshold, which puts more emphasis on confidence rather than sensitivity. Therefore, the resulting F1-score of 76.74% is still fairly strong but is slightly below the optimal balance of 77.37%, which was achieved by the Neural Network.

Of all the models tested, Gradient Boosting achieved the highest level of Precision (85.88%) in classifying locations; thus, when the model classifies a website as being from Bangladesh it is right nearly 86% of the time. However, due to the fact that Gradient Boosting has the lowest Recall (63.18%), this high Precision comes at a cost; therefore, the model's F1-score (72.77%) is an indication of an imbalanced Precision-To-Recall Trade-Off created by the model's "conservative" approach to classifying locations, maximising the model's confidence in its predictions at the expense of broader coverage. Patterns observed during the development process would suggest that the Gradient Boosting model is able to learn sufficient distinctive features that can identify Bangladesh-Origination websites but not enough to be able to generalise to all other instances of positive testing in the dataset tested.

Practical Implications

Trade-off: Precision Vs Recall The precision of 71.89% for the neural network compared to the recall of 70.39% for the random forest translates to approximately 23 fewer false positives per 1,000 predictions and 24 more detected vulnerabilities for every 1,000 actual flaws. An organization will decide to use the option that best suits their resources (precision) or those that are highly critical to security (recall). Geographic Risk Profiling: The accuracy of location prediction (77%–78%) allows

for prioritizing manual review of sites from high-risk geographic areas and enhances tactical vulnerability assessment with more geographic context to allocate resources accordingly. Operational Deployment: The multi-task model's ability to lead in precision makes it the primary model choice for mass automated scanning, where security personnel must triage high-confidence alerts. The single-model dual predictions are computationally efficient and make them suitable for use in environments with limited resources.
By employing a multi-task neural network, researchers were able to predict both vulnerabilities and locations simultaneously with a combined accuracy of 71.89% for vulnerability predictions and 77.37% F1-score for location classifications. The model works best in an operational security setting that places importance on reducing false positives. The 7.5 percentage point advantage in predicting locations as opposed to vulnerabilities illustrates how easy it is to classify into two categories (binary) compared to multiple categories (multi-label) when predicting vulnerabilities. The use of multi-task learning for security functions is very valuable because it allows the ability to create shared representations among different but related tasks, which increases performance, efficiency, and operational value over conventional single-task methods.

SHAP Feature Importance Analysis for Dual Prediction Model

Overview
In this chapter we analyse SHAP (SHapley Additive exPlanations) Feature Importance for the Multi-task Neural Network. The multi-task Neural Network has been used to detect Vulnerabilities within six different groups and predicts the location based on geography. The features that influence the predictions of this model can be determined by using SHAP Values.

5.4.2 Vulnerability Group Feature Importance Analysis

Group Aidentified web_category_TECH (SHAP importance of 0.0145), indicating that injection attack type backends for "Tech" web sites are more complex, other technology-specific backends tend to be less vulnerable. Conversely framework ANGULAR.JS (SHAP importance of 0.0026), indicates minimal impact, since AngularJS is focused primarily on front-end development.
Group B previously identified web_category_TECH (SHAP Importance of 0.0150) as one that represents web platforms that tend to be more complex when it comes to file handling. In addition, server LITESPEED (SHAP importance of 0.0125), was identified with file vulnerabilities, specifically within shared hosting scenarios. Lastly, framework NUXT.JS (SHAP Importance of 0.0012) has very little influence due to its built-in security attributes.
Group C includes the identifying feature backend_language_UNKNOWN (SHAP importance = 0.0150), it highlights the back end of the codebase where the vulnerability may simply result from lack of clarity. Similarly, framework_UNKNOWN (SHAP importance = 0.0135), highlights ambiguous security practices and issues surrounding the protections against XSS and CSRF attacks within Next.js (SHAP import: 0.0020).
Group D has backend_language_PHP (SHAP importance = 0.0131), indicating that PHP based applications are susceptible to authentication flaws and server LITE-

SPEED (SHAP importance = 0.0113), is reflective of their vulnerabilities due to using shared hosts. framework_NUXT.JS (SHAP importance = 0.0020) has a lesser impact on security because of the use of secure, standardized authentication libraries.
In Group E (API and Request Manipulation), the largest impacts from tech websites are indicated by the website category of "Tech" website category (SHAP value = 0.0350), while the Unknown framework (SHAP value = 0.0200), indicates an increased risk of custom APIs. The Laravel framework (SHAP = 0.0025) is almost negligibly impactful, as the default security features of Laravel greatly reduce the vulnerability of custom APIs.
In Group F (Encryption and TLS Issues), Tech websites again lead all groups with a SHAP score of 0.0407 for encryption issues. Technology websites contain a disproportionately high number of vulnerability issues relating to encryption and TLS given the high complexity of technology infrastructure compared to other industries. The Unknown Framework ranks second (SHAP = 0.0239), whereas Other Websites have a negligible risk of encryption and TLS issues (SHAP = 0.0020).

Location Prediction Feature Importance Analysis Websites Coming from Bangladesh are Predicted by the Following Indicators: (1) Server Type (SHAP 0.0465) and (2) Backend Language (PHP) (SHAP 0.0262). These Results Point to the General Trend of Many Bangladeshi Websites Hosting on the Same Servers, as Well as Utilising the Same Backend Technologies.
Websites from Other Countries are Predicted by the Following Indicators: (1) Server Type (SHAP 0.0464), and (2) Backend Language (Unknown) (SHAP 0.0271). These Results show That Many Foreign Websites do Not Provide Any Backend Language Information at All.
SHAP analysis identifies key features across vulnerability and location prediction tasks. Infrastructure features dominate both tasks, while technology-specific categories are more relevant for vulnerability prediction. The architecture's ability to leverage shared features enhances its ability to predict vulnerabilities and geographic origin efficiently.

Confusion Matrix

Group A:
The model achieved an impressive 94.8% accuracy rate by predicting 274 "Safe" instances which were present in Group A testing that included 289 actual "Safe" cases. The 15 instances which were misclassified by the system as "Vulnerable" represented 5.2% of the total. The model demonstrates strong performance in "Safe" instance detection yet it displays an error rate which leads to "Safe" instances being wrongly identified as "Vulnerable" in a minor but detectable portion of cases.
The "Vulnerable" predictions show that the system identified 121 out of 143 cases as "Safe" which corresponds to an 84.6% prediction rate while the system correctly identified 22 cases as "Vulnerable" which represents a 15.4% prediction rate. The model demonstrates a tendency to misclassify "Vulnerable" cases because it prefers to categorize them as "Safe" which results in substantial errors.

Figure 5.11: Confusion Matrix For dual model

Group B:
The model successfully predicted all 427 "Safe" instances without any errors which resulted in 100% accuracy. The model demonstrates its capability to predict "Safe" instances with high accuracy for this particular group.
The model achieved 100% accuracy by predicting all 5 "Vulnerable" instances as "Safe" without making any errors. The model achieved perfect performance by successfully classifying all "Safe" and "Vulnerable" cases in Group B which demonstrates its outstanding classification skills.

Group C:
The model successfully identified 47 of 151 "Safe" cases which represents a success rate of 31.3%. The remaining 103 instances were incorrectly identified by the model as "Vulnerable" which represents 68.7% of the cases. The model demonstrates a significant difficulty which affects its ability to identify "Safe" cases because it fails to identify the majority of "Safe" cases which it treats as "Vulnerable" cases.
The "Vulnerable" category requires that the model correctly predicts 242 out of 282 instances which represents an accuracy rate of 85.8%. The model incorrectly identified 40 instances as "Safe" which represents 14.2% of the total cases. The model demonstrates better performance when detecting "Vulnerable" cases while it faces major difficulties when handling "Safe" cases.

Group D:
Safe Predictions: The model correctly predicted 325 out of 330 actual "Safe" instances (98.2%), with only 6 misclassified as "Vulnerable" (1.8%). This demonstrates that "Safe" instances have been classified with high accuracy.
Vulnerable Predictions: The system achieved correct identification of "Safe" status

in 91 out of 101 "Vulnerable" cases which equals 90.1% accuracy while 10 cases received incorrect identification as "Vulnerable." The "Vulnerable" misclassification rate remains low yet the system still makes significant errors when it designates "Vulnerable" cases as "Safe."

Group E:
Safe Predictions: The model reached a 39.5% success rate in identifying 62 out of 159 "Safe" instances according to its predictions, while 95 instances (60.5%) were misclassified as "Vulnerable." The system shows its main difficulty when it attempts to identify "Safe" cases because it fails to identify more than half of those cases.
Vulnerable Predictions: The system correctly identified 245 out of 335 "Vulnerable" cases, which resulted in an 89.1% accuracy rate, except for 30 cases that the system incorrectly identified as "Safe." The system demonstrates accurate "Vulnerable" predictions, but it fails to deliver correct "Safe" predictions.

Group F:
The model correctly identified 67 of 147 "Safe" instances which were predicted as "Safe" with an accuracy of 45.6% but 80 instances which comprised 54.4% of the total were incorrectly identified as "Vulnerable." The study demonstrates that "Safe" instances suffer from severe misclassification errors because almost half of all instances were wrongly predicted.
The model achieved correct predictions for 254 "Vulnerable" cases out of 285 total cases which resulted in an accuracy rate of 89.1% but 31 cases were incorrectly identified as "Safe." The model demonstrates strong capability to identify "Vulnerable" cases but it still struggles to handle a large number of "Safe" case identifications.

5.5 Comparisons and Relationships

5.5.1 Comparison and Analysis of the Two Models: Vulnerability Prediction vs. Dual Prediction Model

This section compares the performances of two different types of models: The Vulnerability Prediction Model and the Dual Prediction Model. The first model predicts just the vulnerabilities of an item (using 61 input features), while the second model predicts both the vulnerabilities of an item and its likely geographic location (either Bangladesh or not) (using 59 input features). In this section, we will analyze both models against four important metrics: Precision, Recall, F1-Score, and Accuracy.

5.5.2 Vulnerability Prediction:

	Model	Avg Accuracy	Avg Precision	Avg Recall	Avg F1-Score
0	Random Forest	73.6%	56.1%	67.3%	0.599
1	Gradient Boosting	71.9%	53.3%	61.3%	0.557
2	Neural Network	72.1%	54.7%	63.8%	0.566

Figure 5.12: Performance Metrics Comparison

Dual Prediction Model:

Performance Metrics of Dual Model Prediction

Model	Precision	Recall	F1 Score	Accuracy
Neural Network	0.708371 ★	0.721357	0.714805 ★	0.757716 ★
Random Forest	0.695652	0.703941	0.699772	0.745756
Gradient Boosting	0.700976	0.724106 ★	0.712353	0.753858

Figure 5.13: Model-wise Precision, Recall, F1-Score, and Accuracy Comparison

Precision Comparison:

The Dual Model achieves 70.8% accuracy which exceeds the Vulnerability Prediction Model's accuracy of 54.5%. The Dual Model demonstrates better prediction accuracy because it produces fewer false positive results when compared to the Vulnerability Prediction Model.

The Dual Model requires advanced optimization methods together with improved regularization and hyperparameter tuning to achieve better performance. The system uses this method to reduce false positive rates which helps with the vulnerability prediction assessment. The Dual Model's neural network model needs improvements to enhance its ability to generalize when encountering unfamiliar data.

Recall Comparison:
The Dual Model achieves a recall rate of 72.1% which exceeds the 65.9% recall rate of the Vulnerability Prediction Model. The Dual Model demonstrates superior ability to detect real vulnerability sites because its false negative rate remains lower than all other systems. The Dual Model exhibits higher recall results because it detects target classes (vulnerable sites and location) with greater accuracy although this leads to misidentifying some non-vulnerable sites as vulnerable. The Dual Model achieves its optimal performance because it has been designed to handle both tasks at once which creates better results for each individual metric.

F1-Score Comparison:
The Dual Model achieves an F1 Score of 71.5% which exceeds the F1 Score of the Vulnerability Prediction Model that stands at 57.2%. The Dual Model demonstrates superior performance to the Vulnerability Prediction Model because its F1 score measurement combines both precision and recall. The Dual Model shows superior F1 Score performance because it achieves improved precision and recall performance. The model demonstrates enhanced performance because it has been trained to detect both false positive and false negative errors. The Vulnerability Prediction Model exhibits decreased F1 Score results because it suffers from class imbalance and overfitting problems.

Accuracy Comparison:
The Dual Model achieves better accuracy which reaches 75.8% while the Vulnerability Prediction Model shows an accuracy of only 72.3%. The Dual Model demonstrates superior classification abilities because it successfully predicts both vulnerability groups and their corresponding locations. The Dual Model architecture was created to handle multi-label classification of vulnerabilities together with binary classification of locations which enables it to identify intricate patterns in data more effectively. The higher accuracy of the system might result from improved data preprocessing methods which developers optimized for both tasks.

Takeaways and Reasoning for the Improved Performance in the Dual Prediction Model

Multi-Task Learning:
A major factor contributing to the increase in accuracy of the Dual Prediction Model has been its implementation of a multi-task learning framework. This type of framework enables the Dual Prediction Model to simultaneously utilise its training resources to learn common representations of the two tasks it is attempting to predict; ie, the Vulnerability Prediction and the Location Classification tasks. There have been many published results indicating that when two tasks are related, that the dual-use of feature representations across both tasks provides for improved

generalisation capabilities for the model as a whole (Caruana, 1997). When both vulnerabilities and locations are learned simultaneously, all relevant patterns will be captured more completely, allowing the model to produce better results for both vulnerabilities and locations. Caruana (1997) states that performing multi-task learning on related tasks yields higher individual task performance than would be obtained by performing one individual task only, because the model can combine the shared information from both tasks during training to discover new and improved combinations of features, which prevents overfitting and assists in finding the best representation of the features in the training data.

Incorporation of Location Features:
The Dual Prediction Model uses features associated with the physical location of a website's server, the web framework used to build it, and the programming language used to create it. The features describe and differentiate a web server's ENVIRONMENT and TYPE. The additional features help define the website's backend infrastructure and assist the classifier in more accurately predicting WHERE the website is located. An example of this relation is that many web servers can run multiple types of programming languages, which allows the classifier to derive geospatial patterns, based on how each programming language was written and deployed. In past research by Zhang et al., it was shown that by having multiple types of features, the model built using these features was able to perform at a higher level than models created using only one type of feature (technical and contextual).

Better Balance Between Precision and Recall:
Compared to the Vulnerability Prediction Model, the Dual Prediction Model has a higher F1-Score, indicating a more balanced performance in terms of both precision and recall. In other words,. The increase in the Dual Prediction Model's F1Score is likely due to its capacity to learn more robust and generalized representations of both vulnerabilities and locations simultaneously. This is consistent with previous research [19], which found that multi-task learning increases the trade-off between precision and recall for imbalanced datasets while also increasing the accuracy of the model on all tasks. As demonstrated by the studies, multi-task models can leverage shared knowledge across tasks to achieve greater balance across performance metrics, such as precision and recall.

Chapter 6

Conclusion

6.1 Summary of Findings

This research proposed a culturally adaptive neural network framework to analyze cybersecurity vulnerabilities in web applications by learning region specific vulnerability patterns. A custom dataset was developed using vulnerability metadata of web applications vulnerability in Bangladesh, as well as a second dataset of foreign web applications, which allowed the comparative analysis. Bangladeshi data was trained on the neural network to detect the recurring vulnerability structures, which represent localized development practices. The most important observation in this study is that similarity based analysis of foreign web applications can be made by utilizing the vulnerability patterns that were learned using Bangladeshi web applications. In the case of similar patterns of vulnerabilities, the model suggests that there is a potential influence of Bangladeshi developers, which proves that vulnerabilities in cybersecurity can serve as behavioral signatures of development practices. This concludes that machine learning models trained on localized datasets can support culturally informed vulnerability analysis beyond conventional detection approaches.

6.2 Contributions to the Field

In this research, multiple contributions are made to the academic domain of cybersecurity analytics and machine learning, such as the analysis of region-specific vulnerability based on a culturally adaptive framework. Among the main contributions is the development of a fully custom dataset with vulnerability metadata gathered on the Bangladeshi web applications as well as a comparative dataset of the foreign web applications. Since there are no publicly available datasets that could represent vulnerability characteristics affected by culture or region, the dataset in question is significantly lacking and allows doing localized analysis of cybersecurity. Another contribution is the termination of the end to end vulnerability analysis pipeline that was created during this research. The research takes it a step further by classifying the vulnerabilities by the degree of risk where high and low severity are considered vulnerable groups. Methods involving autoencoders were used in order to detect the features of vulnerabilities that are noteworthy, dimensional reduction could be achieved, and meaningful patterns could be observed. Other machine learning methods like gradient boosting were employed to promote predictive

analysis and evaluation metrics like precision and recall ensured good performance measurement.
To improve interpretability and understanding of patterns, various analytical methods were integrated, such as SHAP that was used as a feature importance analysis tool, and TSNE that was used as a vulnerability pattern distribution visualization tool. Vulnerability sequences were also compared using the longest common sequence analysis to determine structural similarity in datasets. Manual intervention was involved in a number of phases, including removing overlap between the framework and the back end and properly adding the location of the developer where the automated solution was not applicable, which further increased data integrity.
The main aspect of this thesis is the creation and deployment of an entirely custom neural network setup that is culturally adaptable in vulnerability prediction. The model was trained on the specifically Bangladeshi data only to be able to learn the recurrent vulnerability patterns linked to the local practices of development. When used with foreign web applications, the model allows a similarity of vulnerability patterns to be used to provide inferences about the possibility of an influence by a Bangladeshi developer. All these contributions show that personalized datasets, explainable machine learning methods, and dedicated neural network models may be integrated to promote culturally sensitive cybersecurity studies.

6.3 Recommendations for Future Work

In future,we will extend this research beyond vulnerability identification toward mitigation by analyzing how vulnerability patterns evolve as developers adapt to new cultural and regional environments.The purpose of this research is to comprehend the impact of cross cultural development on security practices in migration or cross country collaboration of developers, and the impact of the change on the state of vulnerability. This can help in formulating cultural-aware adaptive mitigation strategies.

Bibliography

[1] H. Müller, "Smooth optimum kernel estimators of densities, regression curves and modes," *The Annals of Statistics*, vol. 12, no. 2, pp. 766–774, 1984. DOI: https://doi.org/10.2307/2241411 [Online]. Available: https://www.jstor.org/stable/2241411

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[3] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001. DOI: https://doi.org/10.1214/aos/1013203451

[4] P. Meunier, *Classes of vulnerabilities and attacks*, 2008.

[5] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, Jun. 2011. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[6] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv:1212.5701 [cs]*, Dec. 2012. [Online]. Available: https://arxiv.org/abs/1212.5701

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, Dec. 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[8] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, vol. 1, no. 1, pp. 785–794, Aug. 2016. DOI: https://doi.org/10.1145/2939672.2939785

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, 2016. [Online]. Available: https://www.deeplearningbook.org/

[11] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 1135–1144, Aug. 2016. DOI: https://doi.org/10.1145/2939672.2939778 [Online]. Available: https://arxiv.org/pdf/1602.04938.pdf

[12] C. Catal, A. Akbulut, E. Ekenoglu, and M. Alemdaroglu, "Development of a software vulnerability prediction web service based on artificial neural networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) Workshops: Trends and Applications in Knowledge Discovery and Data Mining*, 2017, pp. 59–67. DOI: 10.1007/978-3-319-67274-8_6

[13] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *arXiv:1705.07874 [cs, stat]*, Nov. 2017. [Online]. Available: https://arxiv.org/abs/1705.07874

[14] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security," in *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2018, pp. 1–8. DOI: 10.1109/CyberSecPODS.2018.8560686

[15] I. Jana and A. Oprea, "Appmine: Behavioral analytics for web application vulnerability detection," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2019, pp. 69–80. DOI: 10.1145/3338466.3358923

[16] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *arXiv preprint arXiv:1906.05799*, 2019.

[17] M. A. Sarker, M. S. Islam, M. M. Alam, and M. M. Alam, "Measuring vulnerabilities of bangladeshi websites," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–6. DOI: 10.1109/ECACE.2019.8679426

[18] S. Sharma, S. Joshi, S. Kumar, and S. Kumar, "Using deep neural networks to translate multi-lingual threat intelligence for cyber security," in *2019 IEEE International Conference on Cyber Security and Privacy (ICSP)*, 2019, pp. 1–6. DOI: 10.1109/ICSP.2019.00011

[19] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[20] F. Nabi, J. Yong, and X. Tao, "Classification of logical vulnerability based on group attacking method," *Procedia Computer Science*, vol. 170, pp. 923–928, 2020.

[21] R. Rabheru, H. Hanif, and S. Maffeis, "A hybrid graph neural network approach for detecting php vulnerabilities," *arXiv preprint arXiv:2012.08835*, 2020.

[22] Z. Akram, M. Majid, and S. Habib, "A systematic literature review: Usage of logistic regression for malware detection," in *2021 International Conference on Innovative Computing (ICIC)*, IEEE, 2021, pp. 1–8.

[23] M. A. A. Hammoudeh, A. Alobaid, A. Alwabli, and F. Alabdulmunim, "The study on assessment of security web applications," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 23, pp. 120–135, Dec. 2021. DOI: 10.3991/ijim.v15i23.27357

[24] L. Wang, R. Abbas, F. M. Almansour, G. S. Gaba, R. Alroobaea, and M. Masud, "An empirical study on vulnerability assessment and penetration detection for highly sensitive networks," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 592–603, 2021. DOI: 10.1515/jisys-2020-0145

[25] S. Ahmed, M. Habibullah, S. Sharmin, N. Ahmed, and K. M. A. Ali, "A study on cyber attacks detection in bangladesh perspective with machine learning," *Preprint*, Mar. 2022. DOI: 10.13140/RG.2.2.18627.68648

[26] R. Alaoui and E. Nfaoui, "Deep learning for vulnerability and attack detection on web applications: A systematic literature review," *Future Internet*, vol. 14, no. 4, p. 118, 2022. DOI: 10.3390/fi14040118

[27] M. M. Alam, M. S. Islam, and M. A. Sarker, "A comprehensive study on cybersecurity threats and defense mechanisms in bangladesh," *Journal of Cybersecurity Research*, vol. 5, no. 2, pp. 123–145, 2023. DOI: 10.1016/j.jcsr.2023.100204

[28] J. H. An, Z. Wang, and I. Joe, "A cnn-based automatic vulnerability detection," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, no. 1, p. 41, 2023. DOI: 10.1186/s13638-023-02255-2

[29] D. Breskuvienė and G. Dzemyda, "Categorical feature encoding techniques for improved classifier performance when dealing with imbalanced data of fraudulent transactions," *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL*, vol. 18, May 2023. DOI: 10.15837/ijccc.2023.3.5433

[30] J. Doe and J. Smith, "Exploring the role of cyber security measures in e-commerce platforms," *SSRN Electronic Journal*, Aug. 2023. DOI: 10.2139/ssrn.4626041

[31] A. Oyetoro, J. Mart, and U. Amah, "Using machine learning techniques random forest and neural network to detect cyber attacks," *ScienceOpen Preprints*, 2023.

[32] F. R. Alzaabi and A. Mehmood, "A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods," *IEEE Access*, vol. 12, pp. 30 907–30 927, 2024.

[33] Z. Aziz and R. Bestak, "Insight into anomaly detection and prediction and mobile network security enhancement leveraging k-means clustering on call detail records," *Sensors*, vol. 24, no. 6, p. 1716, 2024.

[34] A. K. A. Hwaitat and H. N. Fakhouri, "Adaptive cybersecurity neural networks: An evolutionary approach for enhanced attack detection and classification," *Applied Sciences*, vol. 14, no. 19, p. 9142, 2024. DOI: 10.3390/app14199142

[35] R. S. Mim, A. Satter, T. Ahammed, and K. Sakib, "Automated software vulnerability detection using codebert and convolutional neural network," in *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, 2024, pp. 156–167. DOI: 10.5220/0012707900003687

[36] T.-N. Nguyen, N.-S. Vo, D.-T. Le, and K. Nguyen-An, "A study on deep graph neural networks for security vulnerabilities detection in web applications," in *International Conference on Future Data and Security Engineering*, Springer, 2024, pp. 123–137.

[37] D. R. Ojha, "Use of artificial neural networks to detect and prevent cybersecurity threats," *NPRC Journal of Multidisciplinary Research*, vol. 1, no. 6, pp. 132–141, Nov. 2024. DOI: 10.3126/nprcjmr.v1i6.71754

[38] M. Sagar and C. Vanmathi, "A comprehensive review on deep learning techniques on cyber attacks on cyber physical systems," *SN Computer Science*, vol. 5, no. 7, p. 891, 2024.

[39] M. Usman, "Ai-enhanced cybersecurity: Leveraging neural networks for proactive threat detection and prevention," *Preprint*, Nov. 2024. DOI: 10.31224/4065

[40] M. T. Abdelaziz et al., "Enhancing network threat detection with random forest-based nids and permutation feature importance," *Journal of Network and Systems Management*, vol. 33, no. 1, p. 2, 2025.

[41] Z. Liu, "Intelligent classification of computer vulnerabilities and network security management system: Combining memristor neural network and improved tcnn model," *PLOS ONE*, vol. 20, no. 1, e0318075, 2025. DOI: 10.1371/journal.pone.0318075

[42] M. Tanko, A. Bakar Md Sultan, M. Osman, and H. Zulzalil, "An approach for vulnerability detection in web applications using graph neural networks and transformers," *Survey of machine learning models for cybersecurity. Journal of Theoretical and Applied Information Technology (JATIT)*, vol. 103, no. 1, 2025.

[43] C. Singh, V. Vijayalakshmi, and H. Raj, "A machine learning approach for web application vulnerability detection using random forest,"