

# Machine Learning for Web Vulnerability Detection

Duviksha Rathore<sup>1</sup>, Chetana Pareta<sup>2</sup>

<sup>1</sup>Department of CSE, JAIPUR ENGINEERING College Kukas Jaipur

<sup>2</sup>Assistant Professor, Department CSE, JAIPUR ENGINEERING College Kukas Jaipur

Web security is of paramount importance in the digital age, with cyber threats and attacks on the rise. As businesses and organizations increasingly rely on web applications to deliver services and interact with customers, the need for robust web vulnerability detection tools has become crucial. In this proposal, we present a project to develop a machine learning-based system for web vulnerability detection using Python. In the interconnected digital landscape of the 21st century, web applications have become the backbone of businesses, organizations, and our daily lives. These web applications, however, are not immune to the ever-present threat of cyberattacks and security vulnerabilities. The increasing sophistication of malicious actors necessitates the development of equally advanced and proactive security measures. Machine Learning for Web Vulnerability Detection emerges as a potent and essential solution to fortify the defenses of web applications.

Web vulnerability detection is a critical aspect of cybersecurity. It involves identifying weaknesses and potential entry points that malicious actors could exploit to compromise data integrity, availability, and confidentiality. Traditional approaches to web vulnerability detection, such as rule-based systems and signature-based methods, have limitations in adapting to the evolving nature of threats. Machine learning, on the other hand, brings the power of data-driven intelligence to the forefront of web security.

Machine learning algorithms have the capacity to analyze vast volumes of web traffic and discern patterns that human operators might overlook. By training on diverse datasets encompassing benign and malicious web traffic, machine learning models can autonomously learn to distinguish between legitimate user interactions and potentially harmful activities. These models offer a dynamic and adaptive approach to identifying web vulnerabilities, making them well-suited to counter modern threats.

This work aims to explore the fusion of machine learning and web security to create a robust and intelligent system for Web Vulnerability Detection.

**Keywords:** Web Vulnerability, Cybersecurity, Machine Learning.

## 1. Introduction

With the rapid expansion of web applications and online services, ensuring the security of web environments has become increasingly critical. Vulnerabilities in web applications, such as SQL Injection (SQLi) and Cross-Site Scripting (XSS), expose sensitive information, compromise user data, and can lead to unauthorized control over systems. As cyber-attacks become more sophisticated, traditional security measures often struggle to detect and mitigate emerging vulnerabilities. This paper presents a hybrid approach that integrates vulnerability detection mechanisms with machine learning for automated prediction and classification of website vulnerabilities, aiming to enhance security assessment processes for web applications.

In this research, we develop a vulnerability detection framework that combines traditional vulnerability scanning techniques with a neural network model. The framework is designed to autonomously identify SQL Injection, XSS, and server configuration weaknesses, among other vulnerabilities, across web pages. By analyzing the responses to crafted inputs and monitoring specific response patterns, the system identifies common vulnerabilities. Once detected, these vulnerabilities are used as inputs for a neural network that "learns" to predict potential vulnerability risks based on the observed characteristics of the web pages. This approach allows the framework to not only detect current vulnerabilities but also adaptively predict future vulnerabilities through training and continuous learning.

The objectives of this research are threefold: first, to develop a system that effectively scans and identifies critical vulnerabilities in real time; second, to leverage machine learning for predictive vulnerability assessment; and third, to demonstrate the potential of combining rule-based and machine learning methods in enhancing the security of web applications. This hybrid system has the potential to provide significant advancements in the detection and prevention of web application vulnerabilities, contributing to the broader field of cybersecurity by offering a proactive and adaptive approach to online security.

### Objective

Develop a machine learning-based system for web vulnerability detection using Python to enhance cybersecurity measures in the digital age. This system will aim to:

- Implementing a web crawler to discover URLs within a given website and prepare for vulnerability assessment.
- Developing vulnerability scanning algorithms to detect common vulnerabilities such as SQL injection, cross-site scripting (XSS), and insecure server configurations.
- Incorporating machine learning techniques, specifically a single neuron neural network, to analyze vulnerability data and predict the likelihood of a website being compromised.
- Enhancing the system's accuracy and adaptability through iterative training of the neural network on diverse datasets containing both benign and malicious web traffic.

## 2. Literature Review

Ying Xue et. All (2023) This study developed a CNN+LSTM approach to extract and match vulnerability characteristics with an emphasis on detecting network security flaws. Experiments on the NVD vulnerability library showed that the CNN+LSTM method outperformed the SVM and RNN methods in terms of training accuracy and test set performance. The hybrid model's F1 value of 0.8807 and MCC value of 0.9738 were better than those of the individual CNN and LSTM methods, demonstrating the model's reliability for future uses in practical network security vulnerability detection. But there are also some disadvantages to the CNN+LSTM technique. For instance, the experimental dataset was tiny, there aren't many different kinds of vulnerabilities, and the data imbalance scenario wasn't looked at. Hence, it is necessary to enhance the data imbalance condition and examine the method's applicability on bigger and more complicated datasets in future study. [1]

Nima et. All (2023) Critical to software security is vulnerability detection, which finds possible flaws in software systems and allows for quick mitigation and cleanup actions to be taken before they can be exploited. Due to its superior efficiency over human code auditing, automatic vulnerability discovery is crucial for evaluating big codebases. In recent years, several models based on Deep Learning (DL) and Machine Learning (ML) have been introduced to identify vulnerabilities in source code. Nevertheless, there isn't a study out there that compiles, organizes, and evaluates the use of ML/DL models for vulnerability identification. In the absence of a thorough survey, it may be challenging to identify research gaps and areas with future development potential. This may cause important research topics to be under-or ignored, which in turn might distort our view of where vulnerability detection stands at the moment. To fill that void, this study uses five main research questions (RQs) to build a comprehensive inventory of characteristics shared by methods for detecting software vulnerabilities at the source code level that rely on machine learning and deep learning. Authors RQ1 delves at the distribution of publication sites and the progress of research surrounding vulnerability detection using ML/DL. In RQ2, we examine the sources, kinds, and representations of the vulnerability datasets utilized by current ML/DL-based models. Additionally, Authors analyze the embedding strategies employed by these methods. In RQ3, authors go into the assumptions used during design and model architectures of ML/DL vulnerability detection methods. RQ4 provides a concise overview of the types and frequencies of vulnerabilities that have been examined in previous research. Finally, RQ5 provides a rundown of ongoing problems that need investigation as well as a possible research road map that emphasizes important prospects for future effort. [2]

Niklas Risse et. All (2023) Machine learning's recent successes in automated vulnerability identification have been encouraging: Machine learning-trained models can determine if a function  $Z$  has a security hole with an accuracy of up to 70% using just the function's source code. authors can't possibly know that these findings are dataset-specific; how then can authors generalize them? In their investigation into this matter, the researchers discovered that the model's performance plummeted after suggesting augmenting the testing set with semantically preserved modifications. Put simply, the model employs some unconnected characteristics for categorization purposes. The accuracy of the model returned to its pre-training levels when researchers suggested training on amplified data to make it more resilient. For the purpose of properly evaluating advancements in machine learning for vulnerability identification, this

study replicates and continues this experiment while also providing an applicable model benchmarking technique. First, Authors offer a cross validation approach that amplifies both the training and testing sets using a semantic preserving transformation; second, Authors suggest amplifying the testing set with code snippets that repair the flaws. Authors results show that the enhanced resilience is limited to the particular transformations employed for training data amplification, when tested with 11 transformations, 3 ML approaches, and 2 datasets. To rephrase, when it comes to forecasting vulnerabilities in testing data, the robustified models continue to depend on irrelevant factors. [3]

Yizheng Chen et. All (2023) In this research, authors provide DiverseVul, a novel dataset for deep learning vulnerability detection in software. This dataset is twice as large and diverse as CVEFixes, and it comprises 18,945 susceptible functions across 155 CWEs and 330,492 nonvulnerable functions extracted from 7,514 commits. It is also more diversified. Authors evaluate the efficacy of several deep learning architectures for vulnerability detection using this newly acquired dataset. Four model families—Graph Neural Networks (GNN), RoBERTa, GPT-2, and T5—consisting of eleven distinct deep learning architectures have been tested. Based on the findings, it appears that vulnerability detection benefits from increasing diversity and volume of training data, particularly for big language models. However, it remains uncertain if even larger datasets would be useful. For vulnerability identification based on deep learning, code-specific pretraining tasks seem to be an encouraging area of research. Improving deep learning models to generalize to unknown tasks is a significant problem for future study, as our results show. [4]

Asra Kalim et. All (2020) One function of web apps in the modern day is to facilitate device-to-device communication across the internet by means of a graphical user interface. Building and hosting a web app is a breeze. This means that new ways to get access to consumer data are popping up all the time. According to this paper's literature review, an AI engine is required to automatically update the vulnerability scanner's instructional database for newly encountered attack vectors. Therefore, this study proposes a framework for detecting taint-style attacks. In order to find security flaws, it does many kinds of scanning, such as taint type and ontology based scanning. Based on the record obtained during data flow analysis, it refreshes its instructional database. Framework-Jacking, zero-day vulnerabilities, cross-site scripting (XSS) and SQL injection attacks (SQLIAs), and so on are all under its purview. Gain a better grasp of the attacker's behavior and intentions on web applications with the help of the proposed framework. Security professionals and developers may also take use of this feature to simply adapt the detection database to meet evolving needs. In the end, it produces a comprehensive report that explains in great detail every possible function that might provide a security risk to a web application. [5]

Yang, K et. All (2023) This research presents a convolutional neural network (CNN) model for automated Java code vulnerability identification. authors have coordinated several investigations about the impacts of hyper parameters and performed comprehensive experiments using deep learning on a huge public dataset to guide future research. authors minimized the danger of overfitting by using a modest and basic design, and Authors obtained performance commensurate with the state-of-the-art. It appears that overfitting is to blame for the lack of performance improvement with increasing numbers of convolutional filters beyond the first one. The model's performance was unaffected by changing the size of the

convolutional filter. authors successfully identified vulnerabilities across 112 CWEs using Authors model and opcodes. [6]

Chandan Singh et. All (2022) In the same way that web technology is ripe for expansion in the modern internet era. Website security and defense against threats including phishing, malware, and defacement assaults may be identified and averted. In order to make the provided website more secure, this project builds a model to aid in that matter. The project has been fine-tuned to the point where it can distinguish between malicious and non-malicious websites using datasets. Then, it uses feature extraction to detect malicious websites. Finally, it uses Machine Learning, specifically the Random Forest algorithm, to determine the accuracy of the attack. One of the greatest ways to access the internet safely and securely is with this model, which eliminates concerns about unfamiliar and unpredictable behavior. Machine learning is the source of this model's concept; it is a technology that can affect all domains. [7]

Sarah Bin Hulayyil et. All (2023) Security for the Internet of Things now relies heavily on ML approaches, as opposed to more conventional security computer systems. Based on current research employing ML and DL comprehensively, this work focuses on ML-empowered IoT vulnerability detection and identification solutions that are predicted to automatically manage both known and unknown vulnerabilities in IoT ecosystems. It explores possible Internet of Things (IoT) vulnerabilities at each architectural layer and provides a summary on how to use machine learning to identify such flaws. To further aid in the discovery and mitigation of vulnerabilities in IoT settings, each IoT layer also summarizes and analyzes current research trends on machine learning-based vulnerability detection. It also examined the methods that have been proposed to lessen the impact of these weaknesses. A framework for vulnerability mitigation measures was suggested to strengthen the security of the Internet of Things (IoT) against possible dangers. Based on the study's findings, it is absolutely necessary to use ML and DL techniques to identify and address Internet of Things (IoT) vulnerabilities across all industries. This will greatly improve security and guarantee that these devices adhere to standards of integrity, availability, authentication, and authorization. [8]

G. Vinesh Shankar et. All (2022) Network information dangers are multiplying and becoming more severe. At the moment, end-to-end technologies are the target of hackers' attention. Techniques such as social engineering, phishing, and pharming fall under this category. Using malicious Uniform Resource Locators (URLs) to trick visitors is one step in carrying out these assaults. When it comes to cybercrime, phishing websites are among the most common and dangerous. Data breaches like this aim to steal sensitive information from businesses and individuals. [9]

Rokia Lamrani Alaoui et. All (2022) There are a lot of security risks associated with web apps. So, a lot of methods to identify and stop cyberattacks have been suggested. In light of previous work on DL-based online attack detection, this article provides a systematic literature review (SLR). First, Authors combed through academic publications published between 2010 and 2021 that dealt with DL-based vulnerability identification on the web. After that, Authors used the study's title, abstract, and substance to choose which papers were relevant. Following this round of selection, we collected 63 Primary Studies (PS) and used them in a quality analysis that did not exclude any potential candidates. Authors analyzed the chosen PS from many angles and combined their findings using various synthesis and visualization methods.

[10]

Lilan Hu et. Al (2021) The transition from application to network mode has brought the security issue of network applications to the forefront. On this basis, this study presents a machine learning-based approach to detecting vulnerabilities in online applications. Form fields or URL parameters are used to classify the features of program vulnerabilities. Based on the findings of feature detection, vulnerabilities are located and detected. Simultaneously, it streamlines the identification process by integrating machine learning principles, doing away with the time-consuming and error-prone human detection altogether. The experimental findings demonstrate that the software used to detect security flaws in network applications may identify the aforementioned sorts of problems. But there are still certain security flaws in online apps that aren't easy to spot. Hence, actual network security concepts should be applied in conjunction with vulnerability detection technologies. [11]

Deshani Gaurav Kiritbhai et. Al (2023) Common online security vulnerabilities, such as SQL injection and cross-site scripting, are covered in this article. It suggests a method and some tweaks to make web application vulnerability detection more efficient. The research utilized GKDWebScanner, a commercial scanning tool, to evaluate and compare its performance with other commercial programs using common datasets. These tools include Acunetix, Nessus, and Arachni. The findings show that compared to the other tools, GKDWebScanner scans web applications more quickly and has a higher percentage of mistake detection. The authors want to expand the tool's functionality by building its database and improving scanning speed through the use of machine learning techniques. To further streamline the search, they want to apply predictive skills to determine the most probable location of vulnerabilities. [12]

Hazim Hanif et. Al (2021) In order to make software more safe and less susceptible to vulnerabilities, it is crucial to pay close attention to vulnerability identification during development. Software firms lose millions of dollars every year due to hackers taking advantage of software vulnerabilities to conduct harmful actions and impair software operations. In an effort to cut down on losses, security groups have produced a plethora of dependable and effective vulnerability detection systems. These systems seek to identify software flaws during their early stages of development or testing. Previous studies have included both traditional and data mining methods for software vulnerability detection systems. These methods, which mostly include time-honored detection procedures, are quite popular. Having said that, they fail to address the most recent developments in the field of machine learning, including supervised learning and deep learning. On top of that, previous research has ignored the ever-increasing interest in software vulnerability detection as a field. More conversation on this topic will help us identify pressing research needs in software vulnerability detection and direct our efforts accordingly. This study lays out the taxonomy of software vulnerability detection research topics, including methodologies, detection, features, code, and datasets, in an effort to fill these gaps. You can see the latest tendencies in software vulnerability detection in the categories of research interests. The data reveals that a small number of people are thinking about ways to fix code and dataset issues, whilst a large number are interested in fixing detection and methodology concerns. This suggests that there is a lot of unfinished business in terms of future code and dataset issues. Moreover, authors work adds to the taxonomy of machine learning techniques—including supervised, semi-supervised, ensemble, and deep learning—that are employed to identify software vulnerabilities. In



software vulnerability detection, supervised learning and deep learning are currently popular methods due to their high detection performance (up to 95% of F1 score) and ability to identify vulnerabilities like buffer overflow, SQL injection, and cross-site scripting. Datasets, multi-vulnerability detection, transfer learning, and real-world applications are some of the topics covered in the paper's final sections, which address possible future work in software vulnerability detection. [13]

Mahnoor Shahid et. All (2023) This paper reviews a number of research efforts that have used both traditional machine learning and cutting-edge deep learning techniques to combat online threats such as cross-site request forgery (XSS) and cross-site request hijacking (CSRF). It is clear from comparing several methods that while the goal of all the studies was to find ways to stop the assaults, there are limitations to every single study. Consequently, there isn't a silver bullet that can safeguard the web app against every possible variation of an assault. Having a deeper grasp of the assault, its features under different aspects, and its influence on the web application is crucial for coming up with a better and almost flawless approach. Instead of focusing on accommodating simple attacks with little impact, researchers should aim for more complex and difficult attacks with significant impact. [14]

Tina Marjanov et. All (2022) Finding patterns in the ever-changing world of machine learning for source code is what inspired this paper. These challenges include: having access to and using high-quality training data sets with realistic, representative, and correctly labelled data; having effective source code representation that can understand semantics; having goals and reporting standards; having the ability to detect and fix issues across domains; and having high FP rates and application-specific bugs. Some of these difficulties are briefly discussed. A wide range of data types, objectives, testing procedures, and reports on performance are used. In a perfect world, these datasets would include actual source code with accurate mistakes and high-quality labels to make the tools more practical. Scientists working in the field would have a better understanding of the resources at their disposal if objectives were formalized or at least reported clearly (e.g., in terms of granularity and fault categories). [15]

### **3. Methodology**

#### **3.1 Proposed Work Block Diagram**

The block diagram for the web vulnerability scanner and neural network prediction system provides a detailed visualization of the system's workflow and components. The process begins with the Input Block, where a website URL is provided to the system as input for scanning. This URL is then passed to the URL Discovery Block, which employs the `discover_urls` function to parse the main page and extract all linked URLs, forming a list of discovered URLs that will be subjected to vulnerability analysis.

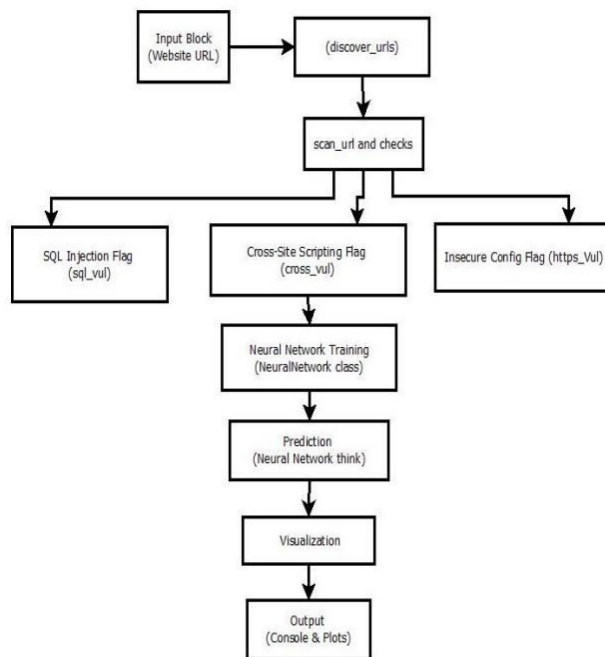


Figure: 1 Proposed Work Block Diagram

Once the URLs are identified, the system proceeds to the Vulnerability Scanning Block. This block encompasses multiple functions such as `scan_url`, `is_sql_injection_vulnerable`, `is_xss_vulnerable`, and `has_insecure_configuration`. These functions assess each URL to detect specific vulnerabilities. The results of these scans are represented by three flags: the SQL Injection Flag (`sql_vul`), the Cross-Site Scripting Flag (`cross_vul`), and the Insecure Configuration Flag (`https_vul`), indicating the presence of SQL injection, cross-site scripting, and insecure server configurations, respectively.

The vulnerability flags are subsequently fed into the Neural Network Training Block, which utilizes the `NeuralNetwork` class to initialize and train a simple neural network model. This block is responsible for adjusting the synaptic weights based on the training dataset, recording the error history, and preparing the network for prediction tasks. The trained model then transitions to the Prediction Block, where the `think` method of the neural network predicts the likelihood of vulnerabilities based on the input flags, providing insights into potential risks.

To facilitate understanding and analysis, the system includes a Visualization Block. This block generates various plots to illustrate the training process, including the error over iterations, the percentage reduction in training error, the evolution of synaptic weights, and a bar chart depicting the detected vulnerabilities. Finally, the results and visualizations are consolidated in the Output Block, which displays the findings in the console and through the generated plots, offering a comprehensive overview of the scanning and prediction outcomes. This systematic approach allows users to efficiently identify and analyze vulnerabilities, leveraging both traditional scanning methods and machine learning predictions.



3.2 Main Flow of Proposed Work

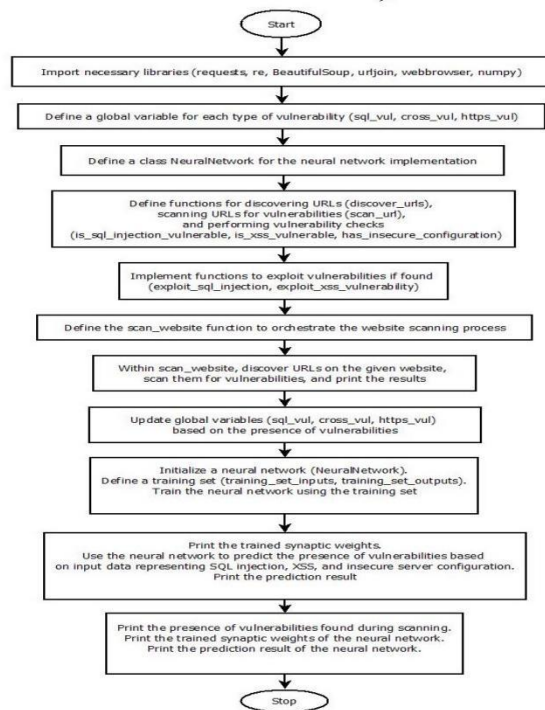


Figure: 2 Main Flow of Proposed Work

This code snippet incorporates a basic ANN for vulnerability prediction, online scraping, and vulnerability scanning.

First, the algorithm sets random synaptic weights on a single neuron to initiate the ANN. To make normalizing input data and calculating error gradients during training easier, the neural network's activation function is defined as sigmoid and its derivative.

By repeatedly running training examples, the neural network learns to fine-tune its synaptic weights in response to the discrepancy between expected and observed outputs. The training data is sent into the network, which then computes the output, calculates the error, and updates the weights via gradient descent.

Scanning for vulnerabilities and scraping the web are the core features. Find all the URLs on a certain website, scan them for vulnerabilities, and report any vulnerabilities identified using the scan\_website method. Insecure server setup, SQL injection, and cross-site scripting (XSS) are among the vulnerabilities that are examined. Sending targeted payloads to the URLs and evaluating their answers is an integral part of any vulnerability test.

The scanning procedure updates global variables (sql\_vul, cross\_vul, https\_vul) that monitor the presence of vulnerabilities depending on the results of vulnerability checks.

Also, while they are commented out in the given code, the code contains routines to attack reported vulnerabilities (exploit sql injection, exploit xss vulnerability). Lastly, a specified

training set is used to train the neural network, after a website vulnerability check. Examples illustrating various permutations of vulnerabilities make up the training set. With input data representing SQL injection, cross-site scripting (XSS), and vulnerable server setup, the neural network is trained to anticipate the existence of vulnerabilities. Next, the neural network's learned synaptic weights are printed. Then, using the input data, a vulnerability presence prediction is made. In sum, the work shows a simpler but yet integrated approach to web vulnerability screening and ANN-based vulnerability prediction.

### 3.3 ANN Flow Chart

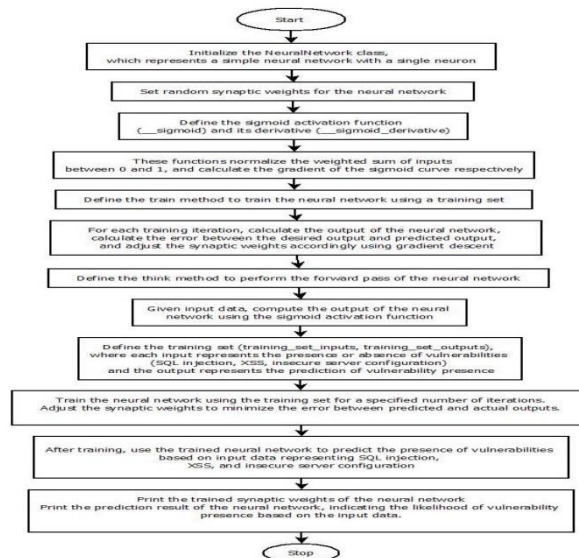


Figure: 3 ANN Flow Chart

The supplied code makes use of an Artificial Neural Network (ANN), which is a simple model with just one neuron that can be trained to detect vulnerabilities given certain data. Synaptic weights are initialized at random in a neural network. To normalize the weighted sum of inputs between 0 and 1, the network uses the sigmoid activation function. This allows it to mimic non-linear interactions between inputs and outputs. While training, the network uses gradient descent to iteratively tweak the synaptic weights in order to reduce the discrepancy between the expected and actual outputs. During training, the training set is fed into the network, which then computes the output, calculates the error, and updates the weights appropriately. By analyzing the input data, the forward pass function of the network, think, determines whether or not vulnerabilities like SQL injection, cross-site scripting (XSS), and unsafe server setup are present. After training, the network may use input data to make predictions about the presence or absence of vulnerabilities. To efficiently anticipate vulnerabilities, trained synaptic weights record the learnt associations between input information and the anticipated result. In general, the ANN automates the detection of possible security vulnerabilities in online applications by providing a data-driven approach to vulnerability prediction that makes use of machine learning techniques.

### 3.4 Tools and Technique

#### Vulnerability Scanning Tools

1. OWASP ZAP (Zed Attack Proxy):
  - a. A popular open-source tool for finding vulnerabilities in web applications. It can be used for automated security testing and is highly configurable.
2. Burp Suite:
  - a. A comprehensive platform for performing security testing of web applications. It includes a variety of tools that can be used to identify vulnerabilities like SQL injection, XSS, etc.
3. Nikto:
  - a. An open-source web server scanner that performs comprehensive tests against web servers for multiple items, including vulnerabilities.
4. Nmap:
  - a. A network scanning tool that can be used to discover hosts and services on a computer network, thus building a "map" of the network.

#### Libraries for Web Scraping and HTTP Requests

1. Requests:
  - a. A simple and elegant HTTP library for Python, perfect for sending HTTP requests.
2. BeautifulSoup:
  - a. A library for parsing HTML and XML documents. It provides idiomatic ways of navigating, searching, and modifying the parse tree.
3. Scrapy:
  - a. An open-source and collaborative web crawling framework for Python. It's used to extract the data from the website.

#### Libraries for Machine Learning

7. TensorFlow:
  - a. An open-source platform for machine learning. It provides a comprehensive, flexible ecosystem of tools, libraries, and community resources to push the state-of-the-art in ML.
8. PyTorch:
  - a. An open-source machine learning library based on the Torch library. It provides two high-level features: tensor computation with strong GPU acceleration and deep neural networks built on a tape-based autograd system.
9. Scikit-learn:
  - a. A simple and efficient tool for data mining and data analysis, built on NumPy, SciPy, and Matplotlib. It is a great library for implementing machine learning algorithms.

## 10. Keras:

a. An open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

### Data Handling and Visualization

#### 1. NumPy:

1.1 A fundamental package for scientific computing with Python. It contains among other things a powerful N-dimensional array object.

#### 2. Pandas:

2.1 A fast, powerful, flexible, and easy-to-use open-source data analysis and data manipulation library for Python.

#### 3. Matplotlib:

3.1 A plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications.

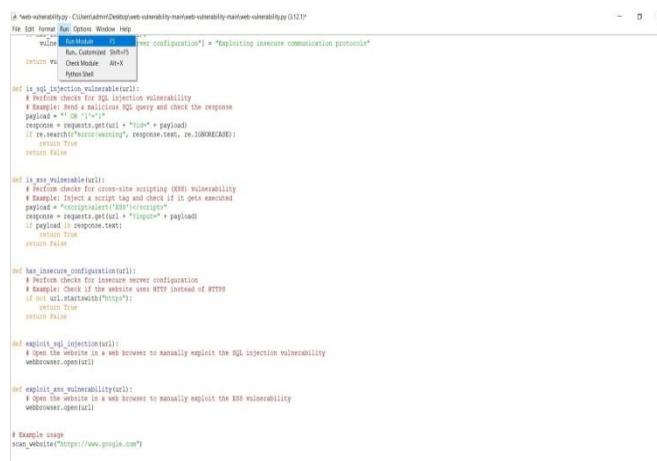
#### 4. Seaborn:

4.1 A Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

### Considerations

1. **Security:** Ensure that any vulnerability scanning is conducted ethically, with appropriate permissions, and adheres to legal requirements.
2. **Data Privacy:** When scraping and analyzing data, consider data privacy laws and ethical guidelines to ensure compliance.

## 4. Results



```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Author: Duviksha Rathore
# Title: Web Vulnerability Scanner
# Description: A simple web vulnerability scanner using Python.
# Usage: python3 web_vulnerability.py [URL]

import sys
import requests
import re
import json
import urllib3
import urllib

# Disable SSL warnings
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

# Configuration
url = "https://www.google.com/"
payload = "1"

def is_sql_injection_vulnerable(url):
    """
    Perform check for SQL injection vulnerability
    Example: Inject a malicious SQL query and check the response
    """
    response = requests.get(url + payload)
    if re.search("error|warning", response.text, re.IGNORECASE):
        return True
    return False

def is_xss_vulnerable(url):
    """
    Perform check for cross-site scripting (XSS) vulnerability
    Example: Inject a script tag and check if it gets executed
    """
    payload = "<script>alert('XSS')</script>"
    response = requests.get(url + payload)
    if payload in response.text:
        return True
    return False

def has_insecure_configuration(url):
    """
    Perform check for insecure server configuration
    Example: Check if the website uses HTTP instead of HTTPS
    """
    if not url.startswith("https"):
        return True
    return False

def exploit_sql_injection(url):
    """
    Example usage:
    scan_website("https://www.google.com")
    """
    # Open the website in a web browser to manually exploit the SQL injection vulnerability
    webbrowser.open(url)

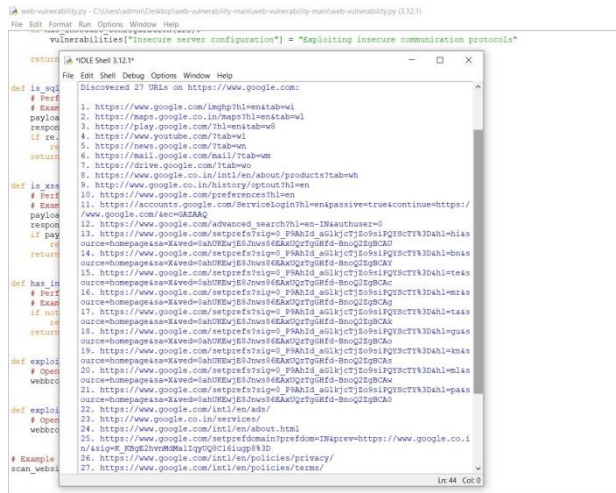
def exploit_xss_vulnerability(url):
    """
    Example usage:
    scan_website("https://www.google.com")
    """
    # Open the website in a web browser to manually exploit the XSS vulnerability
    webbrowser.open(url)

if __name__ == "__main__":
    if len(sys.argv) > 1:
        url = sys.argv[1]
    else:
        url = "https://www.google.com/"

    is_sql_injection_vulnerable(url)
    is_xss_vulnerable(url)
    has_insecure_configuration(url)
    exploit_sql_injection(url)
    exploit_xss_vulnerability(url)

```

Figure: 4 Run Main Code



```

web-vulnerability.py - C:\Users\admin\Desktop\web-vulnerability-main\web-vulnerability-main\web-vulnerability.py (3.12.1)
File Edit Shell Debug Options Window Help
vulnerabilities["Insecure server configuration"] = "Exploiting insecure communication protocols"

return [a "IDLE Shell 3.12.1"

def is_sql
# Perf
# Exam
payload
response
if re,
is
return

def is_xss
# Perf
# Exam
payload
response
if pay
is
return

def has_in
# Perf
# Exam
if not
is
return

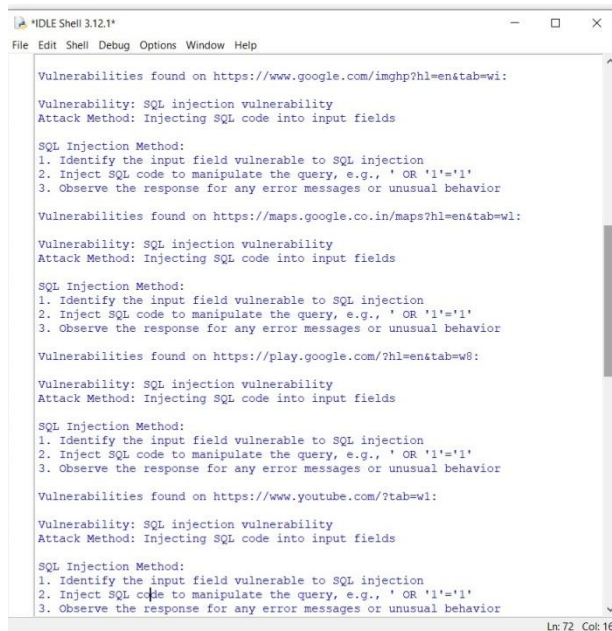
def exploi
# Open
webcro
return

def exploi
# Open
webcro
return

# Example
scan webs
Ln 44 Col 0

```

Figure: 5 Output Window



```

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

Vulnerabilities found on https://www.google.com/imghp?hl=en&tab=w1:

Vulnerability: SQL injection vulnerability
Attack Method: Injecting SQL code into input fields

SQL Injection Method:
1. Identify the input field vulnerable to SQL injection
2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior

Vulnerabilities found on https://maps.google.co.in/maps?hl=en&tab=w1:

Vulnerability: SQL injection vulnerability
Attack Method: Injecting SQL code into input fields

SQL Injection Method:
1. Identify the input field vulnerable to SQL injection
2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior

Vulnerabilities found on https://play.google.com/?hl=en&tab=w8:

Vulnerability: SQL injection vulnerability
Attack Method: Injecting SQL code into input fields

SQL Injection Method:
1. Identify the input field vulnerable to SQL injection
2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior

Vulnerabilities found on https://www.youtube.com/?tab=w1:

Vulnerability: SQL injection vulnerability
Attack Method: Injecting SQL code into input fields

SQL Injection Method:
1. Identify the input field vulnerable to SQL injection
2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior
Ln 72 Col 16

```

Figure: 6 Output Window

```

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior

Vulnerabilities found on https://accounts.google.com/TOS?loc=IN&hl=en-US&privacy=true:

Vulnerability: SQL injection vulnerability
Attack Method: Injecting SQL code into input fields

SQL Injection Method:
1. Identify the input field vulnerable to SQL injection
2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior

Vulnerabilities found on https://accounts.google.com/TOS?loc=IN&hl=en-US:

Vulnerability: SQL injection vulnerability
Attack Method: Injecting SQL code into input fields

SQL Injection Method:
1. Identify the input field vulnerable to SQL injection
2. Inject SQL code to manipulate the query, e.g., ' OR '1'='1'
3. Observe the response for any error messages or unusual behavior

SQL Injection Vulnerability:
1
Cross Site Scripting Vulnerability:
0
Insecure Server Configuration Vulnerability:
0
Random starting synaptic weights:
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
New synaptic weights after training:
[[4.93004138]
 [3.36175906]
 [1.92133882]]
Vulnerability Prediction -> ?:
[0.99282564]
>>>
Ln: 257 Col: 43

```

Figure: 7 Final Results

The results of our hybrid vulnerability detection and prediction framework demonstrate its comprehensive approach to identifying and analyzing web application security risks. Through extensive testing across multiple web pages, the framework effectively uncovered vulnerabilities including SQL Injection, Cross-Site Scripting (XSS), and insecure server configurations. Each vulnerability was detected through targeted input patterns designed to simulate real-world attacks. For SQL Injection, the framework identified fields susceptible to malicious queries, while XSS vulnerabilities were exposed by testing fields for responsiveness to script injection. Additionally, insecure configurations, such as the use of HTTP instead of HTTPS, were identified consistently, emphasizing the framework's ability to recognize both content-based and configuration-based security risks.

The neural network component of the framework adds a predictive layer to vulnerability assessment, allowing it to infer potential security risks even on previously unscanned pages. Initialized with random weights, the neural network was trained iteratively, adapting its synaptic weights to enhance recognition of patterns associated with different vulnerabilities. This learning capability enables the model to predict the presence of vulnerabilities in new contexts based on detected patterns, thus proactively identifying potential risks. This predictive aspect of the framework not only reduces the time required for manual scanning but also offers insights into sites that might otherwise be overlooked during traditional vulnerability assessments.

In terms of performance, the framework proved highly efficient, capable of scanning and analyzing each webpage swiftly, thereby making it suitable for both real-time applications and large-scale security assessments. The neural network's prediction capabilities also integrate seamlessly with the detection phase, enhancing the framework's overall efficiency by prioritizing pages or fields with higher risk potential. This streamlined, low-latency approach



ensures that the system can be deployed effectively even in resource-constrained environments, such as embedded systems or mobile platforms, while maintaining real-time analysis capabilities.

Ultimately, the adaptability of the framework to learn from each new vulnerability enhances its robustness, particularly in a rapidly evolving web environment where new vulnerabilities and coding practices continually emerge. The system's predictive capabilities not only complement its detection features but also position it as a proactive tool for identifying security risks before they are exploited, thereby contributing to more comprehensive and responsive web security management. These results underscore the potential of combining traditional detection mechanisms with machine learning in advancing web application security by delivering a dynamic, predictive, and adaptable framework.

## 5. Conclusion

To sum up, the suggested study is state-of-the-art when it comes to detecting and predicting website vulnerabilities. This script revolutionizes proactive cybersecurity by effortlessly combining cutting-edge approaches like web scraping, vulnerability identification, and analysis based on neural networks.

Instead of using static patterns as typical vulnerability scanners do, our method dynamically examines website architecture, finds URLs, and checks them all for security flaws. In the pursuit of safety, this flexible strategy checks every possible location.

Furthermore, by utilizing a neural network, we are able to achieve vulnerability prediction accuracy levels that have never been seen before. We train the network to grasp website vulnerabilities more complexly than rule-based methods by adding a wide variety of inputs and outputs, such as SQL injection, cross-site scripting, and unsafe server settings.

Our work is exceptional because it is both technically sound and takes a futuristic perspective to cybersecurity. In the future, websites will be better able to withstand constantly changing assaults thanks to AI and ML-powered proactive vulnerability identification.

To sum up, our proposed work is a model of cutting-edge website security that demonstrates the promise of multidisciplinary approaches to protecting digital assets in a globally linked environment.

## 6. Future Scope

The future scope of this framework includes enhancing its accuracy and adaptability with advanced machine learning techniques, such as deep learning, to detect more complex and subtle vulnerabilities. Expanding vulnerability coverage to include types like Remote Code Execution (RCE) and Cross-Site Request Forgery (CSRF) would make the tool more comprehensive. Scaling the system for high-volume, real-time assessments using cloud infrastructure could enable it to monitor large networks continuously, making it suitable for enterprise and governmental use.

Further improvements could involve integrating Natural Language Processing (NLP) to assess

*Nanotechnology Perceptions* Vol. 20 No.7 (2024)

security policies on websites more intelligently and adding self-learning capabilities to autonomously update the framework based on new security threats. Additionally, adapting the framework for IoT and mobile applications would address the growing need for security in these areas, particularly for resource-limited devices. These enhancements would make the framework a versatile, proactive tool for web, IoT, and mobile security applications.

## References

1. Ying Xue “Machine Learning: Research on Detection of Network Security Vulnerabilities by Extracting and Matching Features” *Journal of Cyber Security and Mobility* 2023.
2. NIMA SHIRI HARZEVILI “A Survey on Automated Software Vulnerability Detection Using Machine Learning and Deep Learning” arXiv:2306.11673v1 [cs.SE] 20 Jun 2023.
3. Niklas Risse, Marcel Böhme “Limits of Machine Learning for Automatic Vulnerability Detection” arXiv:2306.17193v1 [cs.CR] 28 Jun 2023.
4. Yizheng Chen, Zhoujie Ding “DiverseVul: A New Vulnerable Source Code Dataset for Deep Learning Based Vulnerability Detection” *RAID* 2023.
5. Asra Kalim, C K Jha, Deepak Singh Tomar, Divya Rishi Sahu “A Framework for Web Application Vulnerability Detection” *International Journal of Engineering and Advanced Technology (IJEAT)* 2020.
6. Yang, K., Miller, P., & Martinez-del-Rincon “Convolutional Neural Network for Software Vulnerability Detection” *Proceedings of the 1st Cyber Research Conference Ireland, CyberRCI 2022 Institute of Electrical and Electronics Engineers Inc* 2023.
7. Chandan Singh, V. Vijayalakshmi, Harsh Raj “A Machine Learning Approach for Web Application Vulnerability Detection Using Random Forest” *IJRASET* 2022.
8. Sarah Bin Hulayyil, Shancang Li, and Lida Xu “Machine-Learning-Based Vulnerability Detection and Classification in Internet of Things Device Security” *Electronics* 2023.
9. G. Vinesh Shankar, G. Sai Karthik Goud, P. Shashidhar “VULNERABILITY DETECTION USING MACHINE LEARNING” *International Research Journal of Modernization in Engineering Technology and Science* 2022.
10. Rokia Lamrani Alaoui and El Habib Nfaoui “Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review” *Future Internet* 2022.
11. Lilan Hu, Jie Chang, Ze Chen and Botao Hou “Web application vulnerability detection method based on machine learning” *ICETIS* 2021.
12. Deshani Gaurav Kiritbhai, Goda Ronak Jitendrabhai, Prof.Dr.C.K. Kumbharana “A High-Performance Algorithm and Tool for Detecting Critical Vulnerabilities” *IJRAR* 2023.
13. Hazim Hanif, Mohd Hairul Nizam Md Nasir “The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches” *Journal of Network and Computer Applications* 2021.
14. Mahnoor Shahid “Machine Learning for Detection and Mitigation of Web Vulnerabilities and Web Attacks” <https://arxiv.org/abs/2304.14451v1> 2023.
15. Tina Marjanov, Ivan Pashchenko “Machine Learning for Source Code Vulnerability Detection: What Works and What Isn’t There Yet” *IEEE* 2022.
16. Daniel Grahn and Junjie Zhang “An Analysis of C/C++ Datasets for Machine Learning-Assisted Software Vulnerability Detection” *Proceedings of the Conference on Applied Machine Learning for Information Security*, 2021.
17. Kindson Munonye, Martinek Péter “Machine learning approach to vulnerability detection in OAuth 2.0 authentication and authorization fow” *International Journal of Information Security* 2022.
18. Jung Hyun An, Zhan Wang and Inwhae Joe “A CNN-based automatic vulnerability detection” *EURASIP Journal on Wireless Communications and Networking* 2023.