

Central Queensland university (CQU) Intellectual property
COIT20265: NETWORK AND INFORMATION SECURITY PROJECT (HT2, 2024)
FINAL REPORT



PROJECT:
GENERATIVE AI: NAVIGATING SHORT-TERM SKEPTISM AND LONG-TERM PROMISE

UNDER THE ESTEEMED GUIDANCE OF

UNIT CO-ORDINATOR: FARIZA SABRINA

MENTOR: DR. AHMEDI AZRA

TEAM MEMBERS:

BASANTA ADHIKARI (12211752)

BHUWAN THAPA (12196590)

KIRAN BHUSAL (12211570)

PRATIK SHINGH DHAMI (12209929)

Table of Contents

1. System Model Diagram.....	3
2. Design of Network / Security Architecture	4
3. Website design and development	6
3.1 Deployment process of WordPress website	8
4. AI Tool to Detect Vulnerabilities in Cybersecurity.....	9
4.1 Software Requirements Specifications (SRS)	10
4.2 Deployment of AI tool onto AWS server	11
4.3 AI to Identify Cybersecurity Vulnerabilities	13
5. List of Issues and Challenges and Mitigation	15
References	17

1. System Model Diagram

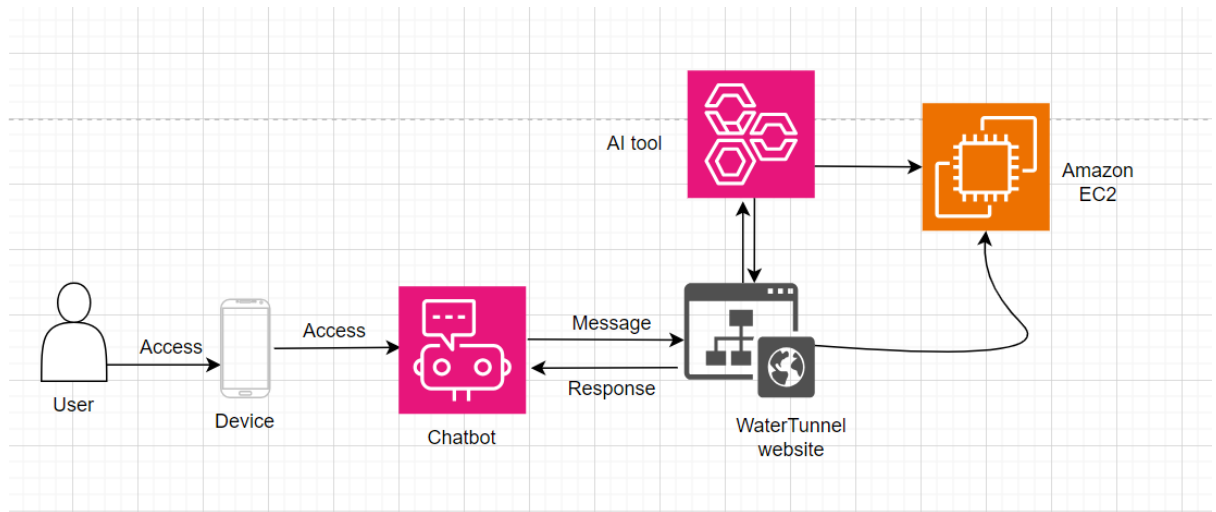


Figure-1: System Design

In the above figure, AI is associated to the website detects the threats and vulnerabilities in the cybersecurity. The website and AI tool both are hosted on EC2 instances under AWS webserver. In addition, while implementing chatbot a user sends an enquiry message through a website, the chatbot will respond to the message from customers. The aim main of the implementing AI tool in this project is to detect cybersecurity vulnerabilities. The AI tool is designed to detect vulnerabilities of the website. The user or security leasers input the URL of the website, AI tool generate vulnerability report that assists and alerts security leaders to make safe secure the company systems. This report will discuss the security proposals for the entire company system just below. All these systems and data is stored and monitored by AWS cloud services.

2. Design of Network / Security Architecture

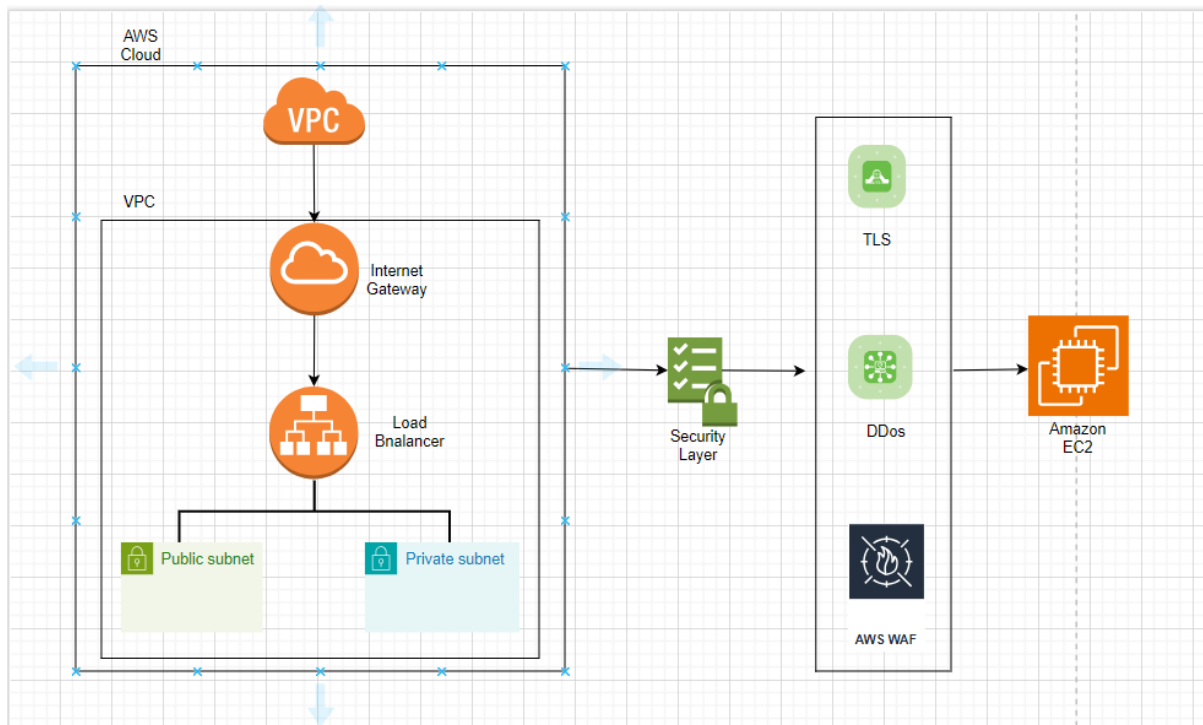


Figure-2: Network/security design

Network Architecture/Protocols/Algorithms

While deploying AI tool and websites on AWS cloud servers, the network architecture plays a crucial role for ensuring secure and efficient communication between clients and servers. Here is a description of the key components of the network architecture:

1. **AWS Cloud:** The entire system is deployed on the AWS cloud infrastructure, which provides scalability, reliability, and a wide range of services to support the deployment of applications and services.
2. **Virtual Private Cloud (VPC):** A VPC is created to logically isolate the network resources of the system within the AWS cloud. It allows to define a virtual network environment with its IP address range, subnets, route tables, and network gateways (Patibandla, K.R., 2024).
3. **Internet Gateway:** An Internet Gateway is attached to the VPC to enable communication between instances in the VPC and the internet. It allows inbound and outbound traffic to and from the internet, facilitating the user's interaction on the website.

4. **Load Balancer:** A load balancer is used to distribute incoming traffic across multiple AWS web servers hosting the website and AI tool. This helps in load distribution, improves availability, and provides fault tolerance by ensuring that no single server is overwhelmed with traffic.
5. **Public Key and Private Key:**
 - Key Infrastructure (PKI) is implemented in the network architecture to secure communications and authenticate entities. “Public keys are used for encryption and verification, while private keys are used for decryption and signing” (Patibandla, K.R., 2024).
 - Public and private key pairs are used for establishing secure connections, such as SSL/TLS connections, between clients and servers. The public key is shared openly, while the private key is kept secure and known only to the server.
 - Public and private keys play a crucial role in ensuring secure communication channels and protecting sensitive data transmitted over the network.

Security Architecture/Protocols/Algorithms

In the deployment of websites and AI tool hosted on AWS cloud servers, a comprehensive security architecture is essential to protect the system from various threats and vulnerabilities. Here is a description of the key components of the security architecture:

1. **Transport Layer Security (TLS):**
 - TLS is implemented to secure communication between clients (e.g. website visitors) and servers (AWS web servers hosting the website).
 - “TLS encrypts data in transit, ensuring that sensitive information exchanged between clients and servers is protected from eavesdropping and tampering” McKay, K. and Cooper, D. (2017, p. 01).
 - By using TLS, the security architecture ensures the confidentiality and integrity of data transmitted over the network.
2. **Distributed Denial of Service (DDoS) Protection:**
 - DDoS protection mechanisms are implemented to defend against DDoS attacks that aim to disrupt the availability of the customers interaction to the website.

- Utilizing AWS Shield, a managed DDoS protection service, helps safeguard the system from volumetric and application layer DDoS attacks by detecting and mitigating malicious traffic.

3. AWS Web Application Firewall (WAF):

- AWS WAF is deployed to protect the web application (website and AI tool) from common web exploits and vulnerabilities (Lakhno et al., 2022).
- It allows security rules to be defined to filter and monitor incoming web traffic, blocking malicious requests before they reach the web servers.
- AWS WAF works in conjunction with the load balancer to provide an additional layer of defence against SQL injection, cross-site scripting (XSS), and other security threats.

with all those above-mentioned network and security architecture, we can enhance security of the website and AI tools. Even tools will be able to notice unethical cyber threats analysing the intension of the users. Including above mentioned security protocols and algorithms and implementing network access control lists within the VPC we can inbound and outbound traffic at the traffic at the subnet level. Configuring ACL we can restrict traffic based on IP addresses and ranges that will help to effectively restrict access from unauthorised access.

3. Website design and development

Regarding the technical progress of the project till now, we have developed and designed a website for Automotive industry specific to WaterTunnel car wash company. For developing websites, deployed Wordpress certified by Bitnami by using AWS server involving several steps. The main reasons for choosing aws cloud service for deploying wordpress are security and cost-effectiveness. When it completed the process of launching instances in EC2 we received a public IPv4 address and login username and password for the wordpress website.

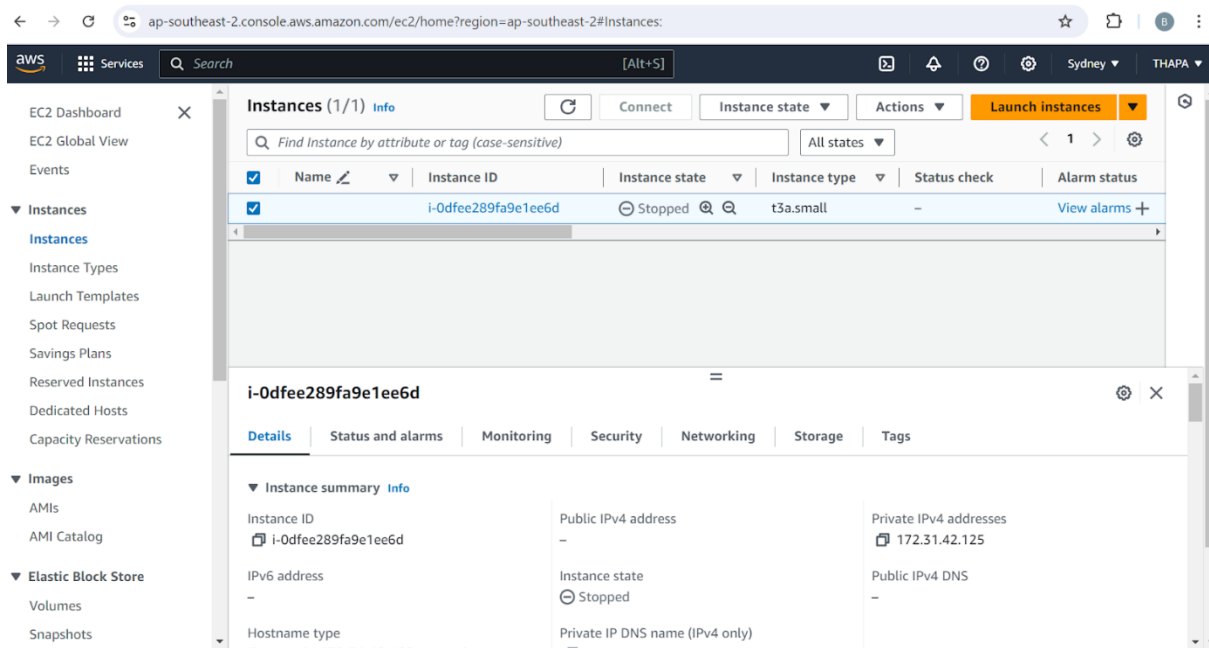


Fig 3: Launching Bitnami WordPress in AWS.

When creating websites then collected different resources from online sources and some of them are self-designed. The website contains everything that needs to be a perfect website. The website is all about company's information and car wash features. The website includes service details, Wash Menu, contact details etc. Here are some glimpses of websites.

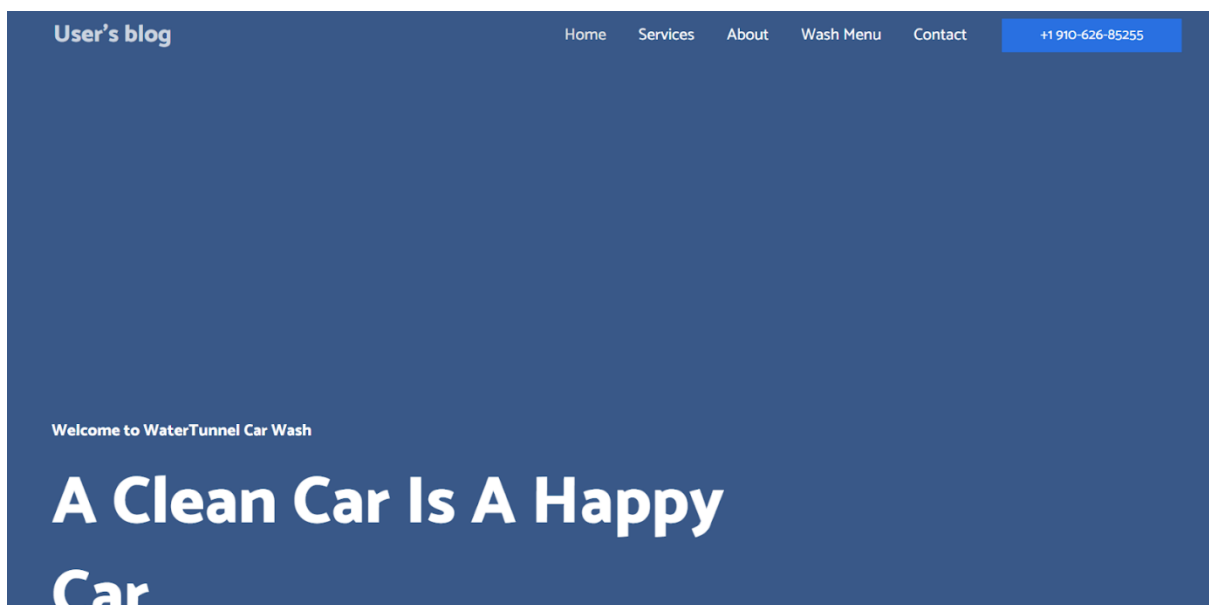


Fig 4: Company Website

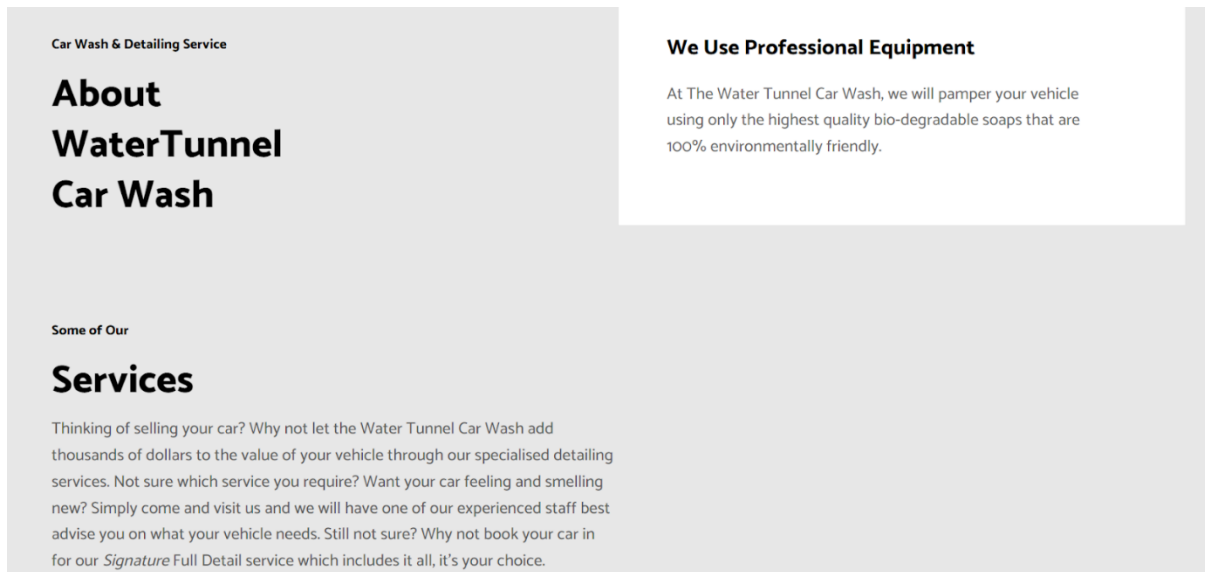


Fig 5: Company Website

3.1 Deployment process of WordPress website

The developed website is deployed into EC2 instance under AWS server for scalability, reliability, security and cost-effectiveness etc.

Deployment process of website in EC2 instance:

1. Login to AWS server.
2. Select “EC2” instances from all services.
3. Select “lunch instance” under EC2.
4. Select “browse more AMIs” by default.
5. Select “AWS Marketplace AMIs” under AMI.
6. Select desired WordPress website.
7. Name website “WordPress or any other name” under key pair.
8. Select “lunch instances” at the bottom and wait for couple of minutes
9. Again, go back to instances under EC2, you will see running instances
10. Select “running instances” you will see details of the website such as public IP and private IP.
11. Copy “public IP” and open in new tab you will see website format and design it.
12. For login username and password for WordPress, select running “instance ID”.
13. Select “action” and again select “monitor and troubleshoot”.
14. Search for user Id and password to login in WordPress site.

4. AI Tool to Detect Vulnerabilities in Cybersecurity

Gen AI plays an important role in providing security layers for the website or any other application. As this report has already discussed about the problems, solutions, benefits and limitations of GenAI while detecting the vulnerabilities in cybersecurity. Here, the aim is to generate such AI tool which detects Vulnerabilities on the website that we already designed.

The AI will be based on the following algorithms:

1. Data collection:
 - Gather website content including HTML, CSS, JavaScript and backend code for analysis.
 - Extract security related metadata such as headers, cookies, and response codes.
2. Input processing:
 - Tokenize and parse the website code into analysable units.
 - Normalise data to eliminate any inconsistencies.
3. Vulnerability detection:
 - Pattern matching: Identify known vulnerability pattern using predefined rule sets (e.g SQL injection, XSS, CSRF).
 - Anomaly detection: Use AI/ML models to detect unusual patterns or deviations in the website's structure and that may indicate new vulnerabilities.
4. Security Compliance Check:
 - Compare website configurations and code against industry security standards (e.g., OWASP Top 10) to ensure compliance.
 - Generate a compliance report highlighting areas of concern.
5. Recommendation Generation:
 - Based on detected vulnerabilities, generate actionable security recommendations.
 - Prioritize vulnerabilities by severity and impact.
6. Reporting:
 - Generate a detailed security report with findings, risk assessments, and recommended actions.
 - Provide options for automatic or manual fixes based on severity.

7. Deployment:

- Integrate the AI tool into a continuous integration/continuous deployment (CI/CD) pipeline.
- Deploy the AI tool on AWS to enable scalability and access control.

8. Feedback Loop:

- Continuously update the AI models with new vulnerability data and improve detection algorithms based on user feedback.

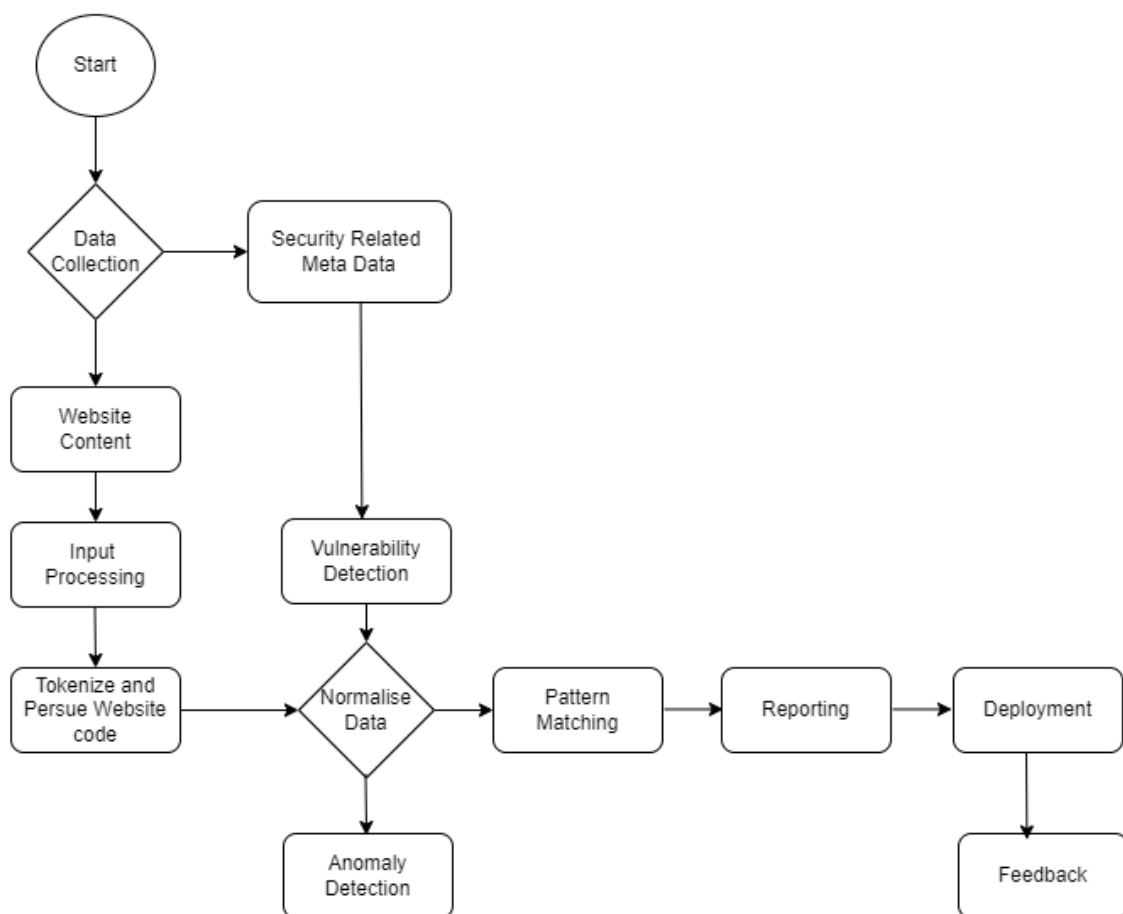


Fig 6: Flowchart for GenAI Tool

4.1 Software Requirements Specifications (SRS)

Here this topic defines the functional and non-functional requirements for the web application vulnerability scanner. The system is developed to scan website for common vulnerabilities like SQL injection, cross-site scripting (XSS), SSRF, RCE and many more. The scanner will help developer and security leaders identify and assess vulnerabilities in their website.

The web application vulnerability will allow user to enter a URL and scan various security vulnerabilities of the website. The system checks for vulnerabilities across common endpoints, Analyse server responses and classify the results. The vulnerabilities logged and reported in the comprehensive format. The scanner also performs brute force login attempts and anomaly detection in server responses.

SQL Injection: A code injection technique that allow unauthorized access to a database, enable attackers to view modify and delete data.

XSS (Cross Site Scripting): When web application accepts user input and includes in webpages without proper validation or sanitization the malicious script is permanently stored in webpages.

Server Site Request Forgery (SSRF): When a user clicks on vulnerable URL, that allows attackers to access the information of the users.

Cross Site Request Forgery (CSRF): An attack where an attackers trapped users to perform actions on web application by sending them links.

Brute force Attack: an attack that used to gain unauthorised access to the system by systematically trying every possible combination of password.

Functional Requirements

1. The system must accept URL as input from user.
2. The system will check common paths such as s (e.g., /wp-admin, /wp-login.php, /phpinfo.php) for vulnerabilities.
3. The system will classify the type of vulnerability using a pre-trained machine learning model such as (e.g., SQL Injection, XSS, CSRF, etc.).
4. The system will attempt brute force login with common username and password combinations.
5. The system will detect anomalies in the server responses based on a trained anomaly detection model.
6. The system will log vulnerabilities and errors to an HTML file.

Interface requirements

1. User interface: valid URL for scanning and HTML pages showing vulnerabilities results, classification and report.
2. No hardware requirements.
3. Software requirements: python, flask that serves as a web framework, TensorFlow used for machine learning classification, Hugging Face Transformers for pre-trained model integration.

4.2 Deployment of AI tool onto AWS server

The developed AI tool is deployed into EC2 instance under AWS server for scalability, reliability, security and cost-effectiveness etc.

Deployment process of AI tool

1. Prepare AI model
 - Choose framework e.g TenssorFlow
2. Create deployment package

- Setup virtual environment

Type: `python -m venv myenv`

`source myenv/bin/activate`

- Install necessary libraries

Type: `pip install tensorflow`

`pip install numpy`

- Package Your Code: Create folder and copy AI model and code

Type: `mkdir my-deployment-package`

`cp -r mymodel.h5 my-deployment-package/`

`cp lambda_function.py my-deployment-package/`

`cd my-deployment-package`

`zip -r ../my-deployment-package.zip .`

`cd ..`

3. Login to AWS management console

4. Navigate to lambda

- Click on "Create function."

- Choose "Author from scratch."

- Fill in the function name, select the runtime (e.g., Python 3.x), and set execution permissions (IAM role).

5. Upload Your Deployment Package

- In the "Function code" section, select "Upload a .zip file" and upload my-deployment-package.zip.

6. Set necessary environment variables in lambda configuration.

7. Increase memory and timeout if necessary.

8. Go to "Test Tab" and create a new test event.

9. Run the test and check the output.

Steps for Running AI tool

1. Login to AWS server.

2. Access the EC2 instance.

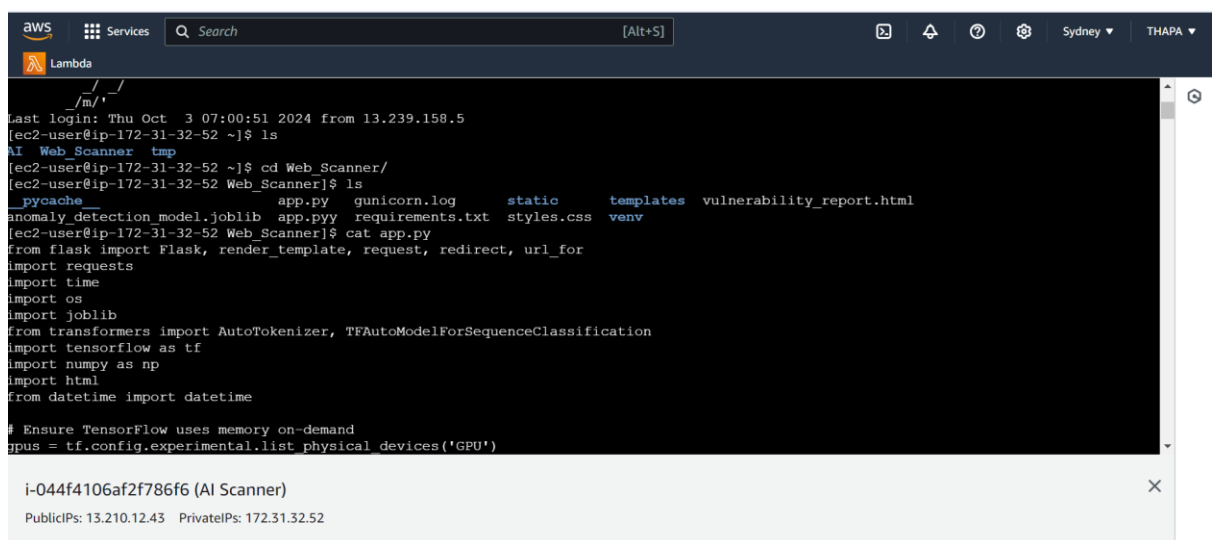
3. Navigate to AI Scanner:

- Select the EC2 instance that host AI Scanner.

4. Start the instance:

- If the instance is in a stopped state, go to the instance state and click start.

5. Connect the EC2 instance:
 - Once the instance is running, select the instance and click connect.
6. Establish a connection:
 - Choose the method to connect to your instance (EC2 instance connect or other) and click connect at the bottom to initiate the session.
7. Navigate to Web Scanner Directory:
 - In the terminal type the following command to navigate to the Web Scanner directory:
cd Web_Scanner/
8. To see deployed python code type: cat app.py
9. Activate the Virtual Environment by typing: source venv/bin/activate
10. Start the Web Server:
 - Start the AI Scanner application using Gunicorn with following command:
gunicorn -w 4 -b 0.0.0.0:5000 -timeout 1200 app:app



```

aws
Services
Search [Alt+S]
Lambda
~/m/
Last login: Thu Oct 3 07:00:51 2024 from 13.239.158.5
[ec2-user@ip-172-31-32-52 ~]$ ls
AI Web_Scanner tmp
[ec2-user@ip-172-31-32-52 ~]$ cd Web_Scanner/
[ec2-user@ip-172-31-32-52 Web_Scanner]$ ls
__pycache__ app.py gunicorn.log static templates vulnerability_report.html
anomaly_detection model.joblib app.py requirements.txt styles.css venv
[ec2-user@ip-172-31-32-52 Web_Scanner]$ cat app.py
from flask import Flask, render_template, request, redirect, url_for
import requests
import time
import os
import joblib
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
import tensorflow as tf
import numpy as np
import html
from datetime import datetime

# Ensure TensorFlow uses memory on-demand
gpus = tf.config.experimental.list_physical_devices('GPU')
  
```

i-044f4106af2f786f6 (AI Scanner)
PublicIPs: 13.210.12.43 PrivateIPs: 172.31.32.52

Fig 7: Running AI Scanner in Virtual Environment

4.3 AI to Identify Cybersecurity Vulnerabilities

This project has completed the basic task of developing AI which is able to detect cybersecurity vulnerabilities, and checked using URL of the website and got positive result. The AI is basically generated in VS code using python programming language and used the basic models available on python library and used AI ML model such as isolation forest and tensor flow for anomaly detection and also used pre trained AI model Microsoft/codebert-base for making our task easy. Basically, the generated AI is able to detect vulnerabilities on the different areas of the websites through SQL injection, Cross site scripting, brute-force attack and we have tried Cross-site Request Forgery (CSRF) and Server-site Request Forgery (SSRF). The result has been shown below.

Vulnerability Scan Report for <http://52.63.41.222/>

Path	Description	Severity Level
wp-admin/install.php	[VULNERABLE] http://52.63.41.222/wp-admin/install.php - WordPress installation script is accessible. (Type: XSS, Confidence: 0.55)	
wp-admin/install.php	[VULNERABLE] SSRF possible at: http://52.63.41.222/wp-admin/install.php with payload: http://localhost [ANOMALOUS RESPONSE DETECTED]	
wp-admin/install.php	[VULNERABLE] SSRF possible at: http://52.63.41.222/wp-admin/install.php with payload: http://127.0.0.1	
wp-admin/install.php	[VULNERABLE] SSRF possible at: http://52.63.41.222/wp-admin/install.php with payload: http://169.254.169.254/latest/meta-data/ [ANOMALOUS RESPONSE DETECTED]	
wp-login.php	[VULNERABLE] http://52.63.41.222/wp-login.php - WordPress login page is exposed. (Type: XSS, Confidence: 0.56)	
wp-login.php	[VULNERABLE] XSS possible at: http://52.63.41.222/wp-login.php with payload: <code><script>alert(0x27;XSS&#x27;)&lt;/script></code>	

Fig 8: Vulnerability Scan Report

wp-login.php	[VULNERABLE] XSS possible at: http://52.63.41.222/wp-login.php with payload:
wp-login.php	[VULNERABLE] XSS possible at: http://52.63.41.222/wp-login.php with payload: <div onmouseover='alert("XSS")'>Hover</div>
wp-login.php	[VULNERABLE] SSRF possible at: http://52.63.41.222/wp-login.php with payload: http://localhost
wp-login.php	[VULNERABLE] SSRF possible at: http://52.63.41.222/wp-login.php with payload: http://127.0.0.1
wp-login.php	[VULNERABLE] SSRF possible at: http://52.63.41.222/wp-login.php with payload: http://169.254.169.254/latest/meta-data/
wp-login.php	[VULNERABLE] RCE possible at: http://52.63.41.222/wp-login.php with payload: phpinfo()
wp-login.php	[VULNERABLE] RCE possible at: http://52.63.41.222/wp-login.php with payload: system("ls")
wp-login.php	[VULNERABLE] RCE possible at: http://52.63.41.222/wp-login.php with payload: system("cat /etc/passwd")
wp-content/debug.log	[VULNERABLE] XSS possible at: http://52.63.41.222/wp-content/debug.log with payload: <script>alert('XSS')</script>

Fig 9: Vulnerability Scan Report

5. List of Issues and Challenges and Mitigation

Issues and Challenges	Mitigation
Finding official WordPress.	For finding official WordPress, went through most of AMI in AWS most of them were paid, chose WordPress certified by Bitnami because of its cost-effectiveness.
Selecting security group while configuration launching.	It confused about whether to select SSH, HTTP or HTTPS. Later, did google research and found out to select all three.
Finding WordPress admin password.	We waited up to 1 hr after completing launch in EC2 to receive username and password.
Unexpected cost.	The cost for launching WordPress will be deducted per hour rate so we must visit AWS web server regular.

While creating the Gantt chart, errors indicated by red lines appeared on the start and end dates of some tasks	We resolved this issue by adjusting the predecessors and re-entering the data, which corrected the errors and allowed us to generate the Gantt chart successfully.
It was difficult to select an appropriate industry and identify the relevant stakeholders.	We chose the automotive industry, specifically a water tunnel car wash. With the help of the manager, we gathered information about the stakeholders.
Scanning urls or multiple tests cases network latency can cause timeouts.	Use reasonable timeout values for 'requests.get' and 'requests.post' that we have used.
Automated vulnerability scanners can produce false positives.	Refined payloads for SQL injection and XSS have been used.
Using an anomaly detection model can yield false positives.	Training data for the anomaly detection model is representative of both normal and abnormal responses.
Training data for the anomaly detection model is representative of both normal and abnormal responses.	Our program includes memory-efficient data structures.
Pre-trained CodeBERT model might not always produce accurate result.	Fine-tune the CodeBERT model with a dataset of labelled vulnerability reports specific to our use case.

References

Patibandla, K.R., 2024. Design and Create VPC in AWS. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, 1(1), pp.273-282.

McKay, K. and Cooper, D., 2017. *Guidelines for the selection, configuration, and use of transport layer security (TLS) implementations* (No. NIST Special Publication (SP) 800-52 Rev. 2 (Draft)). National Institute of Standards and Technology.

Lakhno, V., Blozva, A., Kasatkin, D., Chubaievskyi, V., Shestak, Y., Tyshchenko, D. and Brzhanov, R., 2022. Experimental studies of the features of using waf to protect internal services in the zero trust structure. *J Theor Appl Inf Technol*, 100(3), pp.705-721.