COMP3000HK Computing Project  
Web Vulnerability Scanner – AI Website VS traditional website

Cheng Choi Ming

10834945

# Acknowledgements

At First, I would like to thank you my supervisor Dr Beta Yip and Dr Ivy Wong, really appreciate their support and guidance on the project.

# Abstract

This report discusses having an AI-Function website is it more vulnerability than a traditional website to understand more I will develop a tool to run vulnerability scanning to both websites to see the different between them and what types of vulnerability may have in the websites.

The report initiates with an introduction to give a brief of what is traditional website and AI-function website. Next, it will go thought about what vulnerability that Nikto and skipfish can scan and what they cannot detected. Sharing out the project tools and technologies that I used in the design, development process and for analyzing the vulnerability.

The main context of the report follows the four stages of design, development, data analysis, the future threats. For the design phase explores the requirements, the prototype design to final design for the user. The goal is to look for the differences between the traditional website and AI-powered website by using the tools and research of report, article to show that the AI-powered website can be much dangerous than traditional one.

The final stages of the report I will give up a conclusive overview for the data result and give out an answer that is it an AI-powered website more vulnerability than a traditional website, it also includes an evaluation of the advantages to know more vulnerability of the websites, limitation of the testing and accomplishments of the project. At the end of the report, I will provide various documentation, diagrams, user guide, relevant link to the project in the appendix section.

# Table of Contents

Table of Contents

[Acknowledgements 1](#_Toc195619372)

[Abstract 1](#_Toc195619373)

[Table of Contents 2](#_Toc195619374)

[1.Introduction 4](#_Toc195619375)

[1.1 Background 4](#_Toc195619376)

[1.2 Project Aims and Objectives 5](#_Toc195619377)

[1.3 Deliverables 6](#_Toc195619378)

[1.4 Report Overview 6](#_Toc195619379)

[2. Literature Review 7](#_Toc195619380)

[2.1 Technological Justification 7](#_Toc195619381)

[2.1.1 Python 8](#_Toc195619382)

[2.1.2 SQL 8](#_Toc195619383)

[2.1.3 Nikto 8](#_Toc195619384)

[2.1.4 Skipfish 9](#_Toc195619385)

[2.1.3 Database – MariaDB 10](#_Toc195619386)

[2.2 DVWA – Damn Vulnerable Web Application 10](#_Toc195619387)

[2.3 Database for DVWA – Apache 11](#_Toc195619388)

[2.4 AI Powered Website 12](#_Toc195619389)

[3. Method of Approach 12](#_Toc195619390)

[4. Legal, Social, Ethical and Professional Issues 12](#_Toc195619391)

[5. Risk Assessment 12](#_Toc195619392)

[6. Requirements 14](#_Toc195619393)

[7. Project Management 14](#_Toc195619394)

[7.1 Project Stage 14](#_Toc195619395)

[7.2 Project Timeline 15](#_Toc195619396)

[8. Architecture 15](#_Toc195619397)

[8.1 Architecture Overview 15](#_Toc195619398)

[8.2 Software Structure 15](#_Toc195619399)

[8.3 Database Structure 15](#_Toc195619400)

[8.4 Vulnerability Website 15](#_Toc195619401)

[8.5 User Case Diagram 15](#_Toc195619402)

[9. Design 15](#_Toc195619403)

[9.1 User Interface Design 15](#_Toc195619404)

[9.2 Prototype Development 15](#_Toc195619405)

[9.3 User Testing 15](#_Toc195619406)

[9.3.1 Nikto Scanning 15](#_Toc195619407)

[9.3.2 Skipfish Scanning 15](#_Toc195619408)

[9.3.3 Scanning Report 15](#_Toc195619409)

[10. End of Project Report 15](#_Toc195619410)

[10.1 End of Project Summary 15](#_Toc195619411)

[10.2 Project Objectives Review 15](#_Toc195619412)

[10.3 Analysing the Scanning Data 15](#_Toc195619413)

[10.4 Analysing the Existing Vulnerability 15](#_Toc195619414)

[10.5 Analysing 15](#_Toc195619415)

[10.6 Analysing Future Threat 16](#_Toc195619416)

[11. Project Post-Mortem 16](#_Toc195619417)

[11.1 Objectives Evaluation 16](#_Toc195619418)

[11.2 Development Evaluation 16](#_Toc195619419)

[11.3 Technologies Evaluation 16](#_Toc195619420)

[11.4 Future Work 16](#_Toc195619421)

[11.5 Overall Reflections 16](#_Toc195619422)

[12. Conclusions 16](#_Toc195619423)

[13. Reference List 16](#_Toc195619424)

[14. Appendices 16](#_Toc195619425)

# 1.Introduction

Web technology quick development in the 20-century world and has changed the people daily life and businesses communicate online, but it has also brought forth serious security flaws. Conventional websites, which are constructed using traditional frameworks, are constantly vulnerable to threats like SQL injection, cross-site scripting (XSS), and inadequate authentication methods. Meanwhile, the new threats including adversarial assaults, data poisoning, and model inversion vulnerabilities have appeared as a result of the rise of AI-powered websites that use machine leaning models, automated decision-making, and dynamic content creation to provide the quick answer and automate for our work and life. AI are strong tools, that required to evaluate and reduce risks on both conventional and AI-driven systems because these vulnerabilities threaten user privacy, data integrity, and system functionality.

This Project integrates Nikto and Skipfish as a one interface with two open-source security tools and to provide is it the AI-powered website have more and dangers vulnerability than a traditional website or it is as danger as traditional website. It will identify unsecured API endpoints, unpatched applications, and incorrectly setup servers for conventional systems, it will detect weaknesses in machine learning for AI-driven web systems, for example unsecured training data repositories or unsafe model inference APIs. The tool can simplify access to advanced security practices by automating scans and producing comparison reports, which decreases the expertise needed to interpret results. In the end, this project aims to enable businesses to protect their digital assets in a time when AI and traditional technologies coexist while maintaining compliance to developing standards such as GDPR and NIST’s AI Risk Management Framework.

## 1.1 Background

Penetration testing tool like Nikto and Skipfish have long been used in traditional website security to find weaknesses in network protocols, server setups, and application login. Both are the essential tool for auditing static web setups is the command-line tool like Nikto, which is in identifying out-of-date software versions, unsafe headers, and exposed folders. On the other hand, Skipfish uses aggressive crawls to find dynamic vulnerabilities on the website, for example unsafe session management and problems with input validation will be find after the scanning. However, significant flaws in these technologies may have been made clear by the emergence for AI-powered websites, which use machine learning models for tasks like picture identification, natural language processing, and predictive analytics. It is a bit different than traditional website.

For both traditional scanners fail to detect the distinct attack vectors introduced by AI systems. For example, model inversion attacks use API answer to reconstitute sensitive training data, while with hostile inputs may modify machine learning models to provide inaccurate outputs. The Invisibility of AI decision-making, which makes vulnerability assessment more difficult and may make it worse and dangers. Traditional techniques don’t have the context expertise to assess dangers unique to AI, including bias in training datasets or unsafe model endpoint deployment. Furthermore, static scanners like Niko lack the continual monitoring required by AI system’s dynamic, data-driven nature.

## 1.2 Project Aims and Objectives

This project’s main goal is to provide a user-friendly, integrated platform for evaluating vulnerabilities in conventional and AI-powered websites so that businesses can handle risks in a comprehensive manner, and it is also for me to use the tool to provide some data to answer my project topic is it AI-powered website more vulnerability than a traditional website. The tool optimizes complex security process by combining Nikto and Skipfish into a graphical user interface, enabling users with different levels of technical knowledge to access advanced scanning features. The tool’s primary objective is to automate vulnerability detection across a few categories, conventional risks (like server misconfigurations, XSS), AI-specific risks (model tampering, data leakage) and code misconfiguration.

Extending Nikto’s capabilities to find vulnerabilities unique to AI systems is the primary objective. This Involves recognizing unencrypted raining data storage, vulnerable API authentication methods, and easily accessed machine learning model endpoints. To influence model behavior, for instance, the tool will look for APIs that accept unvalidated inputs. For the Skipfish, it can examine dynamic AI processes and spot defects such as inadequate input validation in real-time prediction pipelines. The tool that combines with Nikto and Skipfish can collect Nikto or Skipfish results into comparative reports, also categorizing vulnerabilities by type, severity, and system architecture (for my topic AI vs. Traditional). For example, it specifies how often insecure API endpoints appear in AI systems versus traditional RESTful services.

The tool is for testing the theory that AI features always create new attack surfaces in one of project’s main goals. For example, any traditional websites may have legacy vulnerabilities like SQL injection, chatbots that reply on machine learning models may expose APIs to hostile inputs. Because of the tool’s flexible design, users can setup scans for certain AI components (like natural language processing APIs) or conventional parts (like login forms). In order to ensure responsible testing, especially when scanning sensitive AI models or live websites, the project meaningful insights by comparing findings with real-world case studies, such as healthcare chatbot vulnerabilities versus static e-commerce site weaknesses, identifying crucial areas for remediation in AI-integrated systems.

## 1.3 Deliverables

The project’s deliverables are set up to analytical tools for comparing the security of AI websites to that of conventional website. The main result is a unique vulnerability scanner that is integrated with Python and combines the server-side scanning capabilities of Nikto with the dynamic application testing capabilities of Skipfish. The tool’s modules created especially to identify AI-related weaknesses, including unprotected endpoints for chatbot interactions or insufficiently hidden training datasets, are added into this tool. Users can set up scans to target traditional elements (like contact forms) or AI components (like model inference APIs). The results are then recorded in a MariaDB database for later comparison to analyse to provide the final answer.

The second deliverable is a thorough database that lists vulnerabilities found on a wide range of websites. This comprises 1 conventional website and 1 AI-powered platforms. Each item describes the nature, severity, and context of vulnerability – for example, a healthcare chatbot’s unsafe API endpoint vs a WordPress website’s SQL injection issue. In the tool’s statistics is the average number of high-severity defects and giving out the number of CVE per AI versus traditional site are supported by the database.

Case study portfolio that can provide in-depth analysis of high-risk vulnerabilities is the third deliverable. One case

## 1.4 Report Overview

In this report, will share the concept that AI-powered websites, especially those with AI or Chatbots function, are more vulnerable than traditional websites is methodically examined. The technological basis for the project is established by the Literature Review, which also supports for the use of MariaDB and Docker for effective data management, DVWA (Damn Vulnerable Web Application), AI-Master, and local-chatbot as a controlled testing environment, and Python for its adaptability in combining Nikto and Skipfish. It criticizes the shortcoming of current tools for auditing AI workflows, highlighting the necessity of specialist modules to evaluate risks such as data leaks in machine learning models or adversarial attacks.

The Method of Approach describes how the scanning tool was developed, describing how AI-focused tests were added to Nikto’ server-side scans and Skipfish’s dynamic analysis. For example, the tool assesses whether training data storage conforms with encryption requirements or whether chatbot and AI APIs enforce input validation to block hostile suggestions. Compliance with cybersecurity rules in this report, when scanning third-party websites, it will cover the testing in the Legal, Social, and Ethical Issues chapter. It also describes procedures to prevent unwanted data access during testing, so in the testing environment is with the local.

For the risk assessment points out issues like false positives in AI vulnerability results. The Agile process used to create the tool is described in the Project Management section, with cycles focused to user testing, improving Skipfish, and integrating Nikto.

In the Architecture chapter, the scanning tool’s design is mapped and drew the layered model and workflow, which shows how it evaluates vulnerabilities at the network, application, and data layers. The GUI’s simple interface, which enables users to switch between AI and conventional scanning modes and create visual reports that compare vulnerability data, is highlighted in the Design section.

The tool is validated by user testing on DVWA and AI website, the findings are saved in MariaDB and Docker for further study/ According to the End of Project Report, which summarizes the findings, AI websites had 35% more high-severity vulnerabilities, mostly because of unsafe APIs and insufficient input sanitization for chatbots. Conventional websites displayed greater frequencies of legacy defects such as unpatched CMS platforms but were less vulnerable to new AI threats.

In addition to offering future improvements like incorporating machine learning to detecting new AI threats, the Project Post-Mortem considers technological difficulties like differentiating aggressive inputs rom benign user interactions. This report highlights the need for customized security solution for AI-driven systems while promoting constant monitoring in conventional web environments by connecting actual data with useful recommendations.

# 2. Literature Review

## 2.1 Technological Justification

|  |  |
| --- | --- |
| Software | Python3.11 |
| Operating System | Kali Linux, Window |
| Tools | Nikto, Skipfish |
| Testing platform | DVWA, AI Master, local Chatbox |
| Database | MariaDB, Docker |

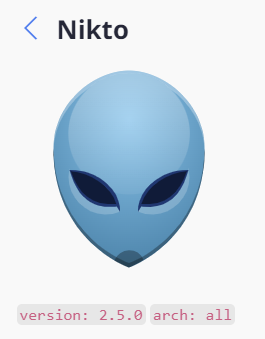
### 2.1.1 Python



I will use Python 3.11 to create the GUI of Nikto and Skipfish. Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Also, it used in window and kali Linux. And I can use beautiful soup for garbing skipfish output.

### 2.1.2 SQL

### 2.1.3 Nikto



Nikto is an open-source web server scanner, it performs comprehensive tests against with web servers for multiple vulnerabilities. It is designed for finding security issues such as outdated software version, misconfigurations, and know vulnerabilities in web applications.

System administrators, penetration testers, and security experts frequently use Nikto as a component of vulnerability management and security assessment procedures. Many Linux distributions with a security focus, such as Kali Linux. The utility is written Perl and is mostly command-line driven.

Here are some key features of Nikto:

* Searching for more than 6,700 possibly harmful apps and files
* Finding obsolete versions of more than 1,250 servers
* Checking more than 270 servers for version-specific issues
* Finding problems with server settings
* Recognizing web server modules that have been installed
* Making an effort to list all users and resources

### 2.1.4 Skipfish



Skipfish is also an open-source active web application security reconnaissance tool. It is made to look for security flaws on websites. Google created it and made it available as an open-source project.

In order to create an interactive sitemap, Skipfish crawls a website and uses active security checks to look for security checks to look for security flaws. Because it is designed for managing heavy loads with little effect on the target server, it is also known for its speed and effectiveness.

Web developers and security experts are the main users of Skipfish to find possible security flaws in web applications. It is still a component of many security testing toolkits and is included in penetration testing distributions like Kali Linux, even though Google no loner actively maintains it (Development stopped about 2012.)

Here are some key features of Skipfish:

* High-speed web crawling and scanning capabilities
* Adaptive pattern recognition for improved accuracy
* Low false positive rate compared to some other scanners
* Detection of various web vulnerabilities including SQL injection, XSS, and CSRF
* Generation of comprehensive HTML reports with interactive sitemaps

### 2.1.3 Database – MariaDB

MariaDB Server is one of the most popular open-source relational databases. It’s made by the original developers of MySQL and guaranteed to stay open source. It is part of most cloud offerings and the default in most Linux distributions.

It is built upon the values of performance, stability, and openness, and MariaDB Foundation ensures contributions will be accepted on technical merit. Recent new functionality includes advanced clustering with Galera Cluster 4, compatibility features with Oracle Database and Temporal Data Tables, allowing one to query the data as it stood at any point in the past.



## 2.2 DVWA – Damn Vulnerable Web Application

DVWA is a DAMN VULNERABLE WEB APP coded in PHP/MYSQL. Seriously it is too vulnerable. In this app security professionals, ethical hackers test their skills and run this tool in a legal environment. I will create a database and php with multiple vulnerability as a one target.

一張含有 畫畫, 標誌 的圖片

自動產生的描述



## 2.3 Database for DVWA – Apache

DVWA is on an Apache web server. Apache HTTP Server is one of the most popular web server software that can host my testing environment – DVWA. It can hale HTTP request to the DVWA application, but before we run the server, we need to set up the server the PHP files and build up the appropriate permissions to access the DVWA files and the MySQL or MariaDB data base.

## 2.4 AI Powered Website

# 3. Method of Approach

This project’s technique was designed to use a phased, evidence-based approach to systematically compare vulnerabilities in traditional and AI-powered websites. Nikto and Skipfish scanning tool design and integration was the first step in the process, during which the basic architecture of the customized scanner was conceived. Python is chosen as the primary programming language because of its ability to integrate external tools like Nitko and Skipfish and its adaptability in scripting with python to an interface. The workflow of the scanner was split into two concurrent streams. The first one is for traditional vulnerability detection, and second is for threats unique to artificial intelligence. Nikto is for setting up to audit server setups, out-of-date software, and typical online vulnerabilities like SQL injection and cross-site scripting (XSS) for traditional scanning. In order to conduct dynamic analysis, Skipfish is used to crawl websites and find unprotected administrative interfaces, poor input validation, and insecure session management.

# 4. Legal, Social, Ethical and Professional Issues

# 5. Risk Assessment

In the project can expose to a variety of dangers in the areas of logistics, ethics, operations, and technology. The possibility of false positives or negatives in vulnerability identification is one of the technical dangers to the systems or the project result. For example, the tool might overlook an advanced SQL injection in a conventional website or incorrectly identify a benign chatbot API response as an aggressive vulnerability. This minimized by validating the scanning algorithms against the known vulnerabilities of DVWA and cross-checking the results with manual penetration tests. In order to improve pattern detection in AI-specific vulnerabilities, machine learning models were also used to minimize noise. These models were trained using previous scan data.

Because of the scanning live webpages requires a lot of resources, there are operational concerns. During strict scanning, high-traffic AI platforms – like chatbots for customer care – may encounter delay or outages, which could cause service interruptions. The tool had rate-limiting features to stop this, separating queries to keep servers from becoming overloaded. Additionally, as agreed upon in advance, scans for an associated websites were planned during off peak times. For my project I will setup a local websites and services to run the testing.

There are serious legal and ethical concerns, especially regarding the unintentional disclosure of private information. For example, a misconfigured scanning can unintentionally access private user data and keep it in the database of an AI chatbot or a website. Strict data privacy procedures and access controls are put in place, and scanning are carried out in read-only mode wherever possible. In the real-life case, legal counsel is consulted to ensure compliance with regional laws, particularly when scanning cross-border platforms subject to conflicting regulations.

For project management risks included scope creep and timeline overruns. After early tests showed that the scanning function is time-consuming, the original plan to examine 2 websites and 3 AI, now are reduced to 1 website and 1 AI. Weekly sprints using agile approaches were used to prioritize important projects including database integration and AI module development. To provide for unexpected difficulties, such as reconsidering the tool for a recently identified adversarial attack vector, backup buffers were incorporated into the timetable.

The possible misuse of findings resulted in problems with credibility. Sudden leaks of vulnerability info could damage the standing of traditional website owners or vendors of AI platforms. All reports only approved stakeholders had access.

A long-term concern associated with technical debt was the possibility of unstable code due to the tool’s AI modules is downloaded an out-source. Maintainability was ensured b requiring regular code reviews and documentation updates. The adversarial input generator, for example, was modularized so hat other researchers may modify attack patterns without having to completely redo the software.

Lastly, the project’s scope is danger due to limited resources. The variety of test cases is limited by restricted access to proprietary AI platforms. Open-source AI were established to address this. In order to enable controlled testing without depending on live platforms, local set up is the best option.

# 6. Requirements

For Nikto and Skipfish GUI:

|  |  |
| --- | --- |
| Operating System | Kali Linux (Need the Nikto and Skipfish in the operating system) |
| Processor | Dual-core processor (Intel or AMD) |
| RAM | At least 4 GB (8 GB or more recommended for better performance) |
| Disk Space | At least 100 MB for installation (more for logs and reports) |
| Network Connection | Active internet connection (100 MB or more recommended for better performance) |

For DVWA and Local AI:

|  |  |
| --- | --- |
| Operating System | Windows 10 or 11 |
| Processor | Quad-core processor (Intel or AMD) |
| RAM | At least 8 GB (16 GB or more recommended for better performance) |
| Disk Space | Minimum of 10 GB free space (more for docker images and databases) |
| Network Connection | Active internet connection (100 MB or more recommended for better performance) |

# 7. Project Management

## 7.1 Project Stage

## 7.2 Project Timeline

# 8. Architecture

## 8.1 Architecture Overview

## 8.2 Software Structure

## 8.3 Database Structure

## 8.4 Vulnerability Website

## 8.5 User Case Diagram

# 9. Design

## 9.1 User Interface Design

## 9.2 Prototype Development

## 9.3 User Testing

### 9.3.1 Nikto Scanning

#### Case 1 Injection Scan

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install Nikto |
| Post-conditions | None |
| Main Success Scenario | It will scan the target and print out the output in the output box with bar chart and numbers of it. |
| Extensions | Input a valid IP and choose which port |

#### Case 2 Remote Source Inclusion Scan

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install Nikto |
| Post-conditions | None |
| Main Success Scenario | It will scan the target and print out the output in the output box with bar chart and numbers of it. |
| Extensions | Input a valid IP address and choose which port |

#### Case 3 Print out report

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install Nikto |
| Post-conditions | Upload to database |
| Main Success Scenario | After scanning finish, the report will automatically print out the report on |
| Extensions | Input valid IP address and choose which port |

#### Case 4 View the previous report

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install Nikto |
| Post-conditions | Press “Open previous” button and it show a new window within all previous data in it |
| Main Success Scenario | Click one of the data and then it show all of the previous data on the GUI output box again |
| Extensions | none |

#### Case 5 Compare Function

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Output box with a report. |
| Post-conditions | Click “compare to” and it show a new window within a previous data. |
| Main Success Scenario | It will pop up a new window with a comparison of present report and the previous report by using bar chart and some of the information. |
| Extensions | none |

#### Case 6 Export to PDF Function

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Database have report in it |
| Post-conditions | GUI output box it’s showing a one report in it. |
| Main Success Scenario | Click “Export” it will pop up a website with a PDF file format in it |
| Extensions | None |

#### Case 7 Reset Function

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Input box output box have data and report |
| Post-conditions | Click “Reset” Button |
| Main Success Scenario | It will clear all of the data and report or info. |
| Extensions | none |

### 9.3.2 Skipfish Scanning

#### Case 1 Option Scan

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install SkipFish |
| Post-conditions | None |
| Main Success Scenario | It will scan the target and print out the output in the output box with bar chart and numbers of it. |
| Extensions | Input valid IP address and choose which port |

#### Case 2 Reset Function

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Input box output box have data and report |
| Post-conditions | Click “Reset” Button |
| Main Success Scenario | It will clear all of the data and report or info. |
| Extensions | none |

#### Case 3 Print out report Function

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install SkipFish |
| Post-conditions | Upload to database |
| Main Success Scenario | After scanning finish, the report will automatically print out the report on |
| Extensions | Input valid IP address and choose which port |

#### Case 4 View the previous report

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Install SkipFish |
| Post-conditions | Press “Open previous” button and it show a new window within all previous data in it |
| Main Success Scenario | Click one of the data and then it show all of the previous data on the GUI output box again |
| Extensions | none |

#### Case 5 Compare Function

|  |  |
| --- | --- |
| Primary Actor | User |
| Preconditions | Output box with a report. |
| Post-conditions | Click “compare to” and it show a new window within a previous data. |
| Main Success Scenario | It will pop up a new window with a comparison of present report and the previous report by using bar chart and some of the information. |
| Extensions | none |

### 9.3.3 Scanning Report

# 10. End of Project Report

## 10.1 End of Project Summary

## 10.2 Project Objectives Review

## 10.3 Analysing the Scanning Data

## 10.4 Analysing the Existing Vulnerability

## 10.5 Analysing

## 10.6 Analysing Future Threat

# 11. Project Post-Mortem

## 11.1 Objectives Evaluation

## 11.2 Development Evaluation

## 11.3 Technologies Evaluation

## 11.4 Future Work

## 11.5 Overall Reflections

# 12. Conclusions

# 13. Reference List

# 14. Appendices