National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

Sourabhkumar Mishra
Student ID: X23300949

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sourabhkumar Mishra
Student ID:	x23300949
Programme:	Msc Cybersecurity **Year:** 2024-25
Module:	Practicum
Lecturer:	Mark Monaghan
Submission Due Date:	11/08/2025
Project Title:	Machine Learning Powered Real Time Web Vulnerability Detection: Browser Extension for Website Security
Word Count:	1599 **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sourabhkumar Mishra
Date:	11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sourabhkumar Mishra
Student ID: x23300949

1 Introduction

1.1 Purpose of this Manual

This document provides a comprehensive guide for the installation, configuration, technical implementation, and usage of the "Machine Learning Powered Real Time Web Vulnerability Detection" browser extension for Microsoft Edge. It details both user-facing features and the underlying code structure that powers the real-time detection engine.

1.2 Product Overview

The Web Vulnerability Scanner is a proactive security tool designed to protect users from common web-based attacks, specifically Cross-Site Scripting (XSS) and SQL Injection (SQLi). It operates directly within the browser, analyzing user inputs and web page content in real-time. The extension leverages the logic derived from a machine learning model trained with 99.88% accuracy, providing instant visual feedback and alerts about potential security threats. All analysis is performed locally, ensuring that user data and browsing habits remain private and secure.

1.3 Intended Audience

This manual is written for a diverse audience:

- **End-Users:** Individuals seeking to enhance their browsing security with a simple, effective tool.
- **Developers & QA Testers:** Professionals who need to understand the extension's internal workings to test, validate, or extend its functionality.
- **System Administrators:** Personnel responsible for deploying security tools within an organization.

2 System Architecture Overview

The system is architecturally divided into two distinct components that work in tandem.

2.1 Offline Machine Learning Pipeline

This is the "brain" of the operation, where the intelligence is created. It is a Python-based process that is run offline to:

1. **Collect and Preprocess Data:** Ingests thousands of samples from public datasets.
2. **Engineer Features:** Extracts over 1000 distinct features (lexical, structural, pattern-based) from each sample.
3. **Train and Evaluate:** Trains a Logistic Regression model to distinguish between malicious and benign payloads with high accuracy.
4. **Generate Artifacts:** Saves the trained model and preprocessing components.

2.2 Online Real-Time Browser Extension

This is the user-facing tool that deploys the "logic" from the trained model. It runs entirely within the user's browser and consists of:

- A **Content Script** that injects into web pages to monitor input fields.
- A **JavaScript Detection Engine** that uses patterns and a scoring system (based on the Python model's findings) to analyze text.
- A **Background Service Worker** to manage notifications and extension state.
- A **Popup UI** for manual control and configuration.

3 System Requirements

3.1 Software Requirements

- **Browser:** A recent version of Microsoft Edge supporting Manifest V3 extensions.
- **Operating System:** Any OS that can run Microsoft Edge (e.g., Windows 10/11, macOS, Linux).

3.2 Required Browser Permissions

The extension requires the following permissions, which are declared in manifest.json. Each is necessary for core functionality:

- **storage:** To save user preferences (e.g., auto-scan enabled) between browsing sessions.
- **notifications:** To display system notifications when a high-risk threat is detected.
- **tabs:** To interact with browser tabs, such as updating the extension's badge with a threat count.
- **scripting:** To inject the content.js and other necessary scripts into web pages for real-time analysis.
- **contextMenus:** To add a "Scan selected text" option to the right-click menu for on-demand scanning.

4 Installation Procedure

The extension is installed by loading it as an "unpacked" extension, which requires enabling developer mode in Edge.

4.1 Step-by-Step Installation Guide

1. **Obtain the Extension Files:** Ensure the complete source code of the extension is located in a single folder on your local machine.
2. **Open Edge Extensions Page:**
    - Launch the Microsoft Edge browser.
    - Navigate to the address edge://extensions by typing it in the address bar and pressing **Enter**.
3. **Enable Developer Mode:**
    - On the Extensions page, find the **"Developer mode"** toggle in the bottom-left corner and switch it **on**. This will enable advanced options for extension management.
4. **Load the Extension:**
    - Click the **"Load unpacked"** button that is now visible at the top of the page.
    - A file dialog will open. Browse to and select the root folder of the extension (the one containing the manifest.json file).
    - Click the **"Select Folder"** button.
5. **Verify Installation:**

- The "Web Vulnerability Scanner" will now appear in your list of installed extensions.
- Its icon should be visible in the Edge toolbar. If not, click the puzzle piece icon (Extensions) and click the pin icon next to the scanner to permanently display it.

The extension is now installed and actively protecting your browser.

5 Core Components & Code Implementation

This section provides a technical breakdown of the key files and code snippets that constitute the browser extension.

5.1 Manifest File (manifest.json)

The manifest.json file is the blueprint of the extension. It tells the browser what the extension does and what permissions it needs.

```
{
 "manifest_version": 3,
 "name": "Web Vulnerability Scanner",
 "version": "1.0",
 "description": "Real-time detection of web vulnerabilities using ML techniques.",
 "permissions": [
  "storage",
  "notifications",
  "tabs",
  "scripting",
  "contextMenus"
 ],
 "background": {
  "service_worker": "background.js"
 },
 "content_scripts": [
  {
   "matches": ["<all_urls>"],
   "js": ["ml-model.js", "content.js"],
   "css": ["styles.css"]
  }
 ],
 "action": {
  "default_popup": "popup.html",
  "default_icon": {
   "16": "icons/icon16.png",
   "48": "icons/icon48.png",
   "128": "icons/icon128.png"
  }
 },
 "icons": {
  "16": "icons/icon16.png",
  "48": "icons/icon48.png",
  "128": "icons/icon128.png"
 }
}
```

- **"manifest_version": 3**: Declares use of the modern, secure Manifest V3 standard.
- **"permissions"**: Lists all required permissions for the extension to function.

- **"background"**: Registers background.js as the persistent service worker.
- **"content_scripts"**: Instructs Edge to inject ml-model.js, content.js, and styles.css into all web pages (<all_urls>).
- **"action"**: Defines the popup HTML file and the extension's toolbar icons.

5.2 Background Service Worker (background.js)

This script manages global extension events and state.

- **onInstalled listener**: Sets default user settings in chrome.storage the first time the extension is installed.
- **onUpdated listener**: The mechanism for auto-scanning. It waits for a page to completely load, checks if auto-scan is enabled, and then injects a scanning function.

5.3 Client-Side Detection Engine (ml-model.js)

This is the JavaScript implementation of the detection logic. It uses patterns and scoring to approximate the Python model.

- **xssPatterns & sqlPatterns**: Arrays of regular expressions that capture the most indicative signs of XSS and SQLi attacks.
- **predictVulnerability(payload)**: This core method calculates a score based on how many dangerous patterns are found in the input payload. It returns an object with the prediction and a confidence score.

5.4 Real-Time Content Monitor (content.js)

This script is injected into every web page to monitor for user input and apply visual feedback.

- **monitorInputs()**: Selects all relevant input fields on the page.
- **addEventListener**: Attaches listeners for input (typing) and paste events to each field.
- **handleInput(element)**: This function is called on every event. It gets the input's value, passes it to the detector, and then dynamically adds the appropriate CSS class (vuln-scanner-danger, etc.) to the element to change its border color.
- **MutationObserver**: Ensures that the extension also monitors input fields created by JavaScript after the page has loaded (common in modern single-page applications).

5.5 Popup User Interface (popup.html & popup.js)

- The HTML provides the structure for the buttons, text areas, and checkboxes.
- The popup.js script adds functionality. The example shows the event listener for the "Test" button, which takes the text from the textarea, runs it through the local VulnerabilityDetector, and displays the result directly in the popup.

6 User Guide & Configuration

This section details the day-to-day usage of the extension.

6.1 The Popup Interface

Click the extension icon in the Edge toolbar to access all manual controls.

6.2 Real-Time Monitoring

This is the default, passive protection mode. As you type into forms on any website, the border of the input field will change color to reflect the assessed risk level:

- 🟢 **Green Border:** Safe input.
- 🟡 **Yellow Border:** Potentially suspicious input.

- 🔴 **Red Border:** High-risk input detected. A system notification will also appear if enabled.

6.3 Manual Page Scanning

To scan an entire page at once:

1. Navigate to the desired web page.
2. Click the extension icon.
3. Click the **"Scan Current Page"** button.
4. The extension will analyze the page, and the popup will update to show any detected threats.

6.4 Payload Analyzer

To test any piece of text:

1. Open the extension popup.
2. Enter the text into the **"Test Payload"** text area.
3. Click the **"Test"** button. The result will be displayed immediately below.

6.5 Configuring Settings

In the popup, click the "Settings" dropdown:

- **Auto-scan:** Enabled by default. Uncheck to turn off all real-time monitoring.
- **Show notifications:** Enabled by default. Uncheck to stop receiving system pop-up notifications for high-risk threats.

7 Validation and Testing

Use the following inputs to verify the extension is functioning correctly.

7.1 Testing Benign Inputs (Expected: Green Border)

- Welcome to our website
- Search for product information
- Contact us at support@example.com

7.2 Testing XSS Payloads (Expected: Red Border & Notification)

- <script>alert('XSS')</script>
- <img src=x onerror=alert('XSS')>
- javascript:alert(document.cookie)
- %3Cscript%3Ealert('XSS')%3C/script%3E (Encoded)

7.3 Testing SQL Injection Payloads (Expected: Red Border & Notification)

- ' OR '1'='1' --
- ' UNION SELECT user, pass FROM users --
- admin'; DROP TABLE users; --

8 Troubleshooting Guide

- **Issue:** Extension not working or icon not visible.
  - **Solution:** Go to edge://extensions. Ensure "Developer mode" is on and the extension toggle is on. Pin the extension to your toolbar from the puzzle piece menu.
- **Issue:** No threats are being detected.

    - **Solution:** Open the popup, go to "Settings," and ensure "Auto-scan" is checked. Reload the page.
- **Issue:** False positives on developer sites (e.g., GitHub).
    - **Solution:** The extension correctly identifies code patterns that match attacks. Use your judgment on trusted sites. You can temporarily disable the extension for a specific site by right-clicking its icon.

9 Conclusion

This manual has provided a complete guide to the configuration and technical implementation of the Web Vulnerability Scanner. By following these instructions, users and testers can effectively install, use, and validate this powerful security tool. The extension provides a critical layer of real-time, client-side protection, enhancing user safety across the web in a privacy-preserving manner.