National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

Nishanth Madasamy
Student ID: x23396369

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nishanth Madasamy
Student ID:	X23396369
Programme:	MSc Cybersecurity **Year:** 2025
Module:	Practicum
Lecturer:	Khadija Hafeez
Submission Due Date:	11/12/2025
Project Title:	Automated Detection of web and Cloud Misconfigurations

Word Count: .....1333..... **Page Count:** .....10.....

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.

NISHANTH MADASAMY

Signature: .....

Date: .....

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

Nishanth Madasamy
X23396369

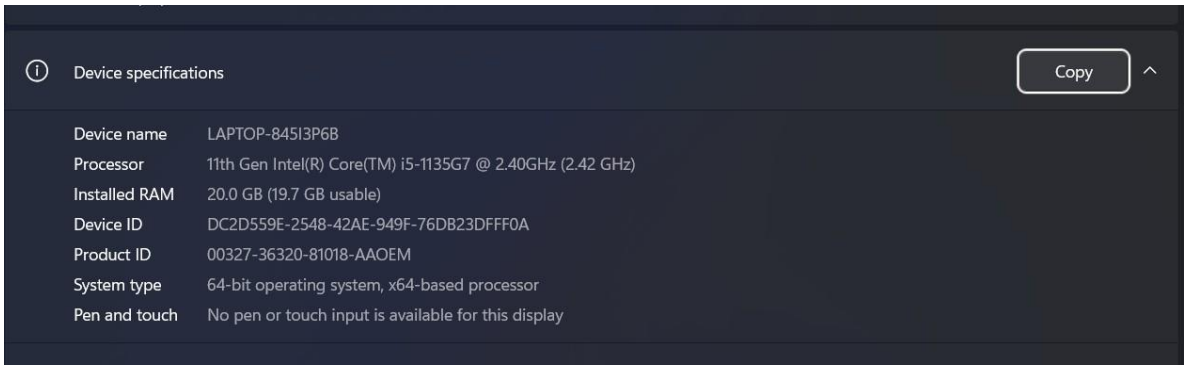
1 Introduction

This Configuration Manual is a detailed guide on setting up and installing the Unified Misconfiguration Detection Suite developed as part of the MSc Cybersecurity Practicum Project and how to run it. The purpose of this document is to make sure that the system can be replicated, executed and validated in a consistent way across different environments by both academic staff, supervisors and technical evaluators. The unified scanner is made of three independent security analysis modules (CORS Misconfiguration Scanner, S3 Bucket Exposure Scanner and OAuth/OIDC Misconfiguration Scanner) merged into one CLI-based unified toolkit. Each module conducts specific probing of misconfigurations in the real world that are common in modern web applications and in the cloud. To achieve a smooth setting up and reproducibility, this manual describes all the hardware, software and environment configurations needed to operate the system. The manual also contains information on how to install Python dependencies, set up vulnerable test environments, run each scanner, and how to interpret the results of output, to demonstrate or video-based evaluation. By following this guide, users will be able to deploy the tool with reliability, reproduce the results obtained during the development of the project, and assess the performance of the scanner in multiple misconfiguration scenarios.

2 Device Specification

This section describes the hardware and virtual machine setup that was used in the development and testing of the Unified Web Misconfiguration Detection Suite. The project was implemented and executed in a controlled environment using Kali Linux virtual environment to ensure consistency, repeatability and isolation from production systems.

2.1 Hardware Specification for the development device



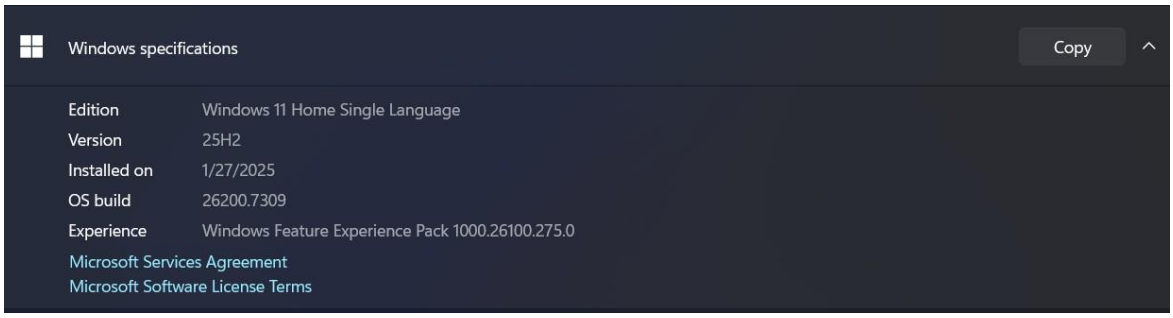


Figure 1 Hardware Specification for the development device

2.2 Hardware Specification for the development device

The Unified Web Misconfiguration Detection Suite development was performed on a Kali Linux virtual machine installed on Oracle VirtualBox.

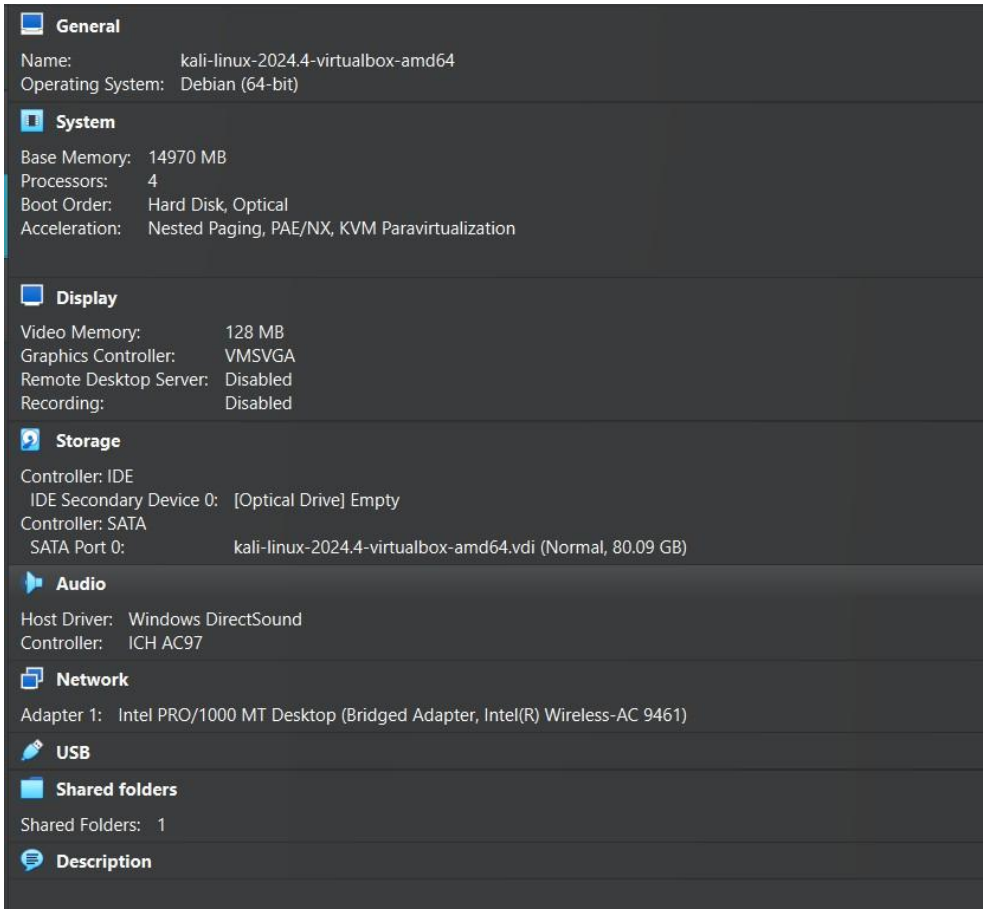


Figure 2 hardware requirement for the deployment environment

The VM was set up with enough RAM (15 GB) and four processors and it was a stable environment to run python-based scanners, local servers, and testing tools. The storage capacity of 80GB ensured that there is enough space to install dependencies, save logs, and host vulnerable test environments (Flask server, MinIO S3 instance, OAuth test server). This hardware setup is sufficient for developing and testing the web vulnerability scanners, and for network-based testing and running local cloud simulation environments without any performance problems.

3 Installation and setup

3.1 Install Virtual Machine (kali Linux)

1. Download the latest kali Linux ISO file from the official website
2. Install the Oracle VirtualBox
3. Create a new VM
4. Attach the ISO and complete the OS installation
5. Install VirtualBox Guest Additions

3.2 Update System Packages

Run the following commands in the terminal of kali Linux

1. Update the local package by this command

```
$ sudo apt update
```

2. Upgrade all installed packages to their latest version

```
$ sudo apt upgrade
```

3. Install the python 3

```
$ sudo apt install python3
```

4. Install PIP packages

```
$ sudo apt install python3-pip
```

5. Install Virtual Environment Module

```
$ sudo apt install python3-venv
```

6. Create the project directory

```
$ mkdir web-scanner (any name we can keep)
```

7. Create and activate the virtual environment

```
$python3 -m venv venv
```

8. Activate the virtual environment by this command

```
$ source venv/bin/activate
```

9. Install the required python libraries to run the code.

```
$ pip install requests
$ pip install urllib3
$ pip install certify
$ pip install beautifulsoup4
$ pip install rich
$ pip install boto3
$pip install flask
```

10. Navigate to the directory we create before

```
$ cd web-scanner
```

11. Create a python file and paste the code (Note : have to create four separate python file in a single folder)

```
$ nano cors.py (detects cors vulnerability)
$nano oauth_vuln_scanner.py (oauth vulnerability scanner)
$nano dsb.py (s3 bucket scanner)
$nano web_suite_fin.py (unified scanner of all three)
$nano vuln_app.py (vulnerable web app)
```

12. Run the code by the below command

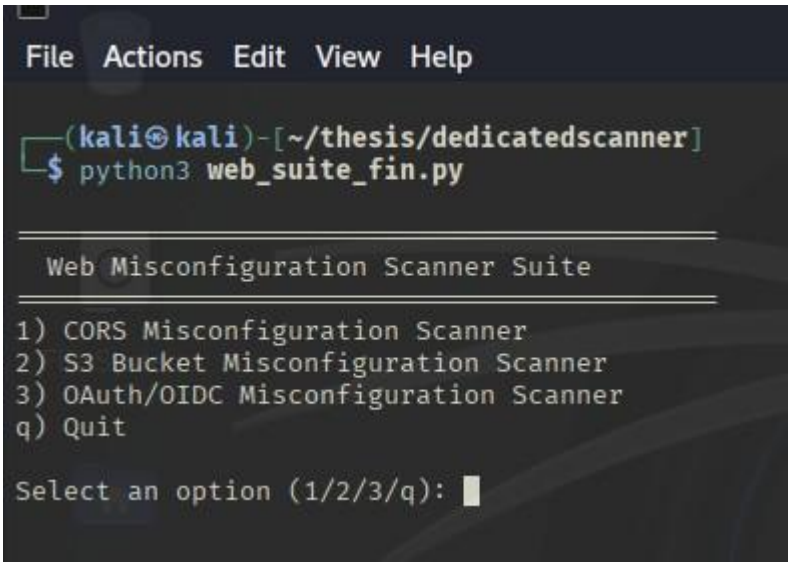


Figure 3 run the code using python3 web_suite_fin.py

4 Scanner Implementation

This section will describe how three scanners (CORS, S3, OAuth) and the unified CLI suite were implemented, installed, configured, and executed.

This project consists of a unified misconfiguration scanning suite that integrates three independent modules:

- CORS Misconfiguration Scanner
- S3 bucket Misconfiguration Scanner
- OAuth/OIDC Misconfiguration Scanner
- Unified CLI Tool

4.1 Python libraries

The system requires the following libraries

- Requests → Performs HTTP
- Boto3 → S3-like communication
- Urllib3 → Error handling and Suppression
- Json, argparse, re → Parsing, CLI, and pattern matching
- Rich → Pretty terminal output

4.2 CORS Scanner Implementation

The COR Scanner Sends Origin-Manipulated requests to detect:

- Reflected origin behaviour
- Wildcard Behaviour
- Wildcard + Credentials issue
- Unsafe preflight responses
- Missing CORS headers

The implementation includes:

Request Handler
Sends GET and OPTIONS requests with a custom origin header.

Header Analysis Logic

Parses:

- Access-Control-Allow-Origin
- Access-Control-Allow-Credentials
- Access-Control-Allow-Methods

Output Formatting
Results are displayed using **rich** tables for clarity.

4.3 S3 Bucket Misconfiguration Scanner(dsb.py)

The S3 Scanner performs anonymous access testing like attacker behaviour .it implements:

Public Read Test
Attempts to **GET** objects anonymously

Public Write Test
Uploads a safe test file to verify write exposure

List Permission Test
Tries unauthenticated list types to check if objects can be listed publicly

4.4 OAUTH/OIDC Scanner Implementation (oauth_vuln_scanner.py)

This scanner detects the common OAUTH/OIDC misconfigurations

Metadata discovery
Fetches:

- /.well-known/openid-configuration
- authorization_endpoint
- token_endpoint
- jwks_uri

Security Checks
Detects:

- Missing PKCE
- Implicit Flow enabled
- Missing JWKS
- HTTP endpoints instead of HTTPS
- Token strings inside JS Files
- Hardcoded client secrets

JavaScript Scanner
Reads every JS file linked on the homepage to detect:

- Client_id
- Client_secret
- Redirect_uri
- Access token inside code

4.5 Unified CLI Tool(web_suite_fin.py)

This script integrates all the three scanners into one user-friendly tool
Features

- Menu-driven interface
- Calls Scanner based on user selection
- Passes URL, flags, and options to each module
- Displays formatted output

Execution flow

- User selects scanner (1/2/3)
- CLI takes required output
- Calls respective script
- Displays results in terminal

References

1. docs.aws.amazon.com. (n.d.). *What is Amazon S3? - Amazon Simple Storage Service*. [online] Available at: https://docs.aws.amazon.com/AmazonS3/latest/userguide//Welcome.html.

2. Flask (2010). *Welcome to Flask — Flask Documentation (3.0.x)*. [online] Palletsprojects.com. Available at: https://flask.palletsprojects.com/en/stable/.

3. OAuth (2020). *OAuth 2.0*. [online] Oauth.net. Available at: https://oauth.net/2/.

4. Pypa.io. (2008). *pip - The Python Package Installer — pip 19.0.3 documentation*. [online] Available at: https://pip.pypa.io/en/stable/.

5. Python Software Foundation (2019). *venv — Creation of virtual environments — Python 3.8.1 documentation*. [online] Python.org. Available at: https://docs.python.org/3/library/venv.html.

6. Reitz, K. (2024). *Requests: HTTP for Humans™ — Requests 2.27.1 Documentation*. [online] requests.readthedocs.io. Available at: https://requests.readthedocs.io/en/latest/.