NEXUS INTELLIGENCE

AI-Powered Automated Cyber Reconnaissance & Audit System **Developed by:** Nexus Team

Project Idea

Overview

NEXUS is an advanced cybersecurity intelligence tool designed to automate the **Reconnaissance (Recon)** phase of penetration testing. Unlike traditional command-line tools, NEXUS provides a unified, graphical interface that leverages Artificial Intelligence logic to map attack surfaces, identify technologies, and assess security risks in real-time.

Problem Statement

Security analysts and bug hunters often spend 70% of their time manually gathering information (subdomains, open ports, server types) using disjointed tools. This process is time-consuming and prone to human error.

The Solution

NEXUS acts as a centralized "Cyber Brain" that:

1. **Visualizes** network topology using interactive graphs.
2. **Automates** vulnerability scanning using AI heuristics.
3. **Generates** professional audit reports (PDF) for stakeholders.
4. **Prioritizes** targets based on calculated risk scores.

2. Algorithm(s) Used

The system integrates four distinct AI and algorithmic modules to process data intelligently:

A. Subdomain Spider (Network Mapping)

- **Algorithm: Greedy Best-First Search**.
- **Description:** The spider traverses the network graph looking for subdomains. It uses a **Heuristic Function** to assign higher weights to sensitive names (e.g., admin, dev, api). The algorithm prioritizes resolving and visualizing these high-value nodes first, ensuring the analyst sees critical targets immediately.

B. Smart Port Scanner (Resource Optimization)

- **Algorithm: Heuristic Priority-Based Search**.
- **Description:** Instead of a brute-force linear scan (1-65535), the scanner utilizes a **Priority Queue** based on a Knowledge Base of common attack vectors. It scans high-probability ports (e.g., 80, 443, 3389) first. This reduces scan time by approximately 90% compared to traditional methods.

C. Tech Fingerprinter (Identification)

- **Algorithm: Rule-Based Expert System**.
- **Description:** The module acts as an inference engine. It applies a set of **If-Then rules** to HTTP headers, cookies, and HTML content to deduce the underlying technology stack

(e.g., IF cookie contains 'laravel_session' THEN Framework = Laravel).

D. Risk Assessor (Decision Support)

- **Algorithm: Weighted Scoring Model (Decision Support System)**.
- **Description:** The system aggregates findings from all modules and applies a mathematical model to calculate a **Cumulative Risk Score (0-100)**.
  - *Equation:* $Risk = \sum (Vulnerability\_Weight \times Impact\_Factor)$
  - Example: Missing X-Frame-Options adds 20 points; Open Port 22 adds 15 points.

3. System Design Outline

Architecture Pattern

The project follows a **Client-Server Architecture** decoupled via a RESTful API.

A. Backend (The Brain)

- **Language:** Python 3.
- **Framework:** Flask (Micro-framework).
- **Responsibilities:**
  - Handling HTTP Requests.
  - Running Scanning Engines (Socket & Requests libraries).
  - Generating PDF Reports (ReportLab).
  - Serving JSON responses.

B. Frontend (The Face)

- **Framework:** Flutter (Dart).
- **Responsibilities:**
  - User Interface (Dashboard, Settings, Graphs).
  - Data Visualization (Force-Directed Graph).
  - State Management.
  - Cross-platform deployment (Linux, Windows, Android).

C. Data Flow Diagram

1. **Input:** User enters Domain (e.g., example.com).
2. **Processing:** Flutter sends GET request to Python Backend.
3. **Analysis:** Python runs AI Algorithms (Spider/Scanner/Fingerprint).
4. **Response:** Python returns JSON data.
5. **Visualization:** Flutter renders the results into Cards and Graphs.
6. **Reporting:** User requests PDF -> Python generates file -> Frontend downloads it.

4. How to Run the Project

Prerequisites

- **Python 3.x** installed.
- **Flutter SDK** installed (Version 3.10+).
- **Linux/Windows/macOS** environment.

Step 1: Backend Setup

1. Navigate to the project folder.
2. Install Python dependencies:

3. Bash
4. pip install flask flask_cors reportlab requests
5. Run the server:
6. Bash
7. python3 app.py
8. *(Output: Running on http://127.0.0.1:5000)*

Step 2: Frontend Setup

1. Open a new terminal in the project folder.
2. Install Flutter packages:
3. Bash
4. flutter pub get
5. Run the application:
6. Bash
7. flutter run -d linux

5. Input/Output Samples

Sample 1: Dashboard Scan

- **Input:** eelu.edu.eg
- **Output (JSON):**
- JSON
- {
- "technologies": ["Apache", "PHP", "Moodle"],
- "risk_score": 65,
- "risk_level": "Medium",
- "open_ports": [80, 443, 8080]
- }
- **Visual Output:** Dashboard displays "Medium Risk" badge, Tech Stack cards show "Apache", and the Spider graph plots subdomains.

Sample 2: PDF Report Generation

- **Input:** Click "EXPORT REPORT" button.
- **Output:** A downloadable file named vortex_report_eelu.edu.eg.pdf.
- **Content:**
  - **Header:** "NEXUS SECURITY AUDIT".
  - **Section 1:** Risk Score (65/100).
  - **Section 2:** Tech Stack (Apache/PHP).
  - **Section 3:** List of Open Ports.
  - **Footer:** "Generated by NEXUS Engine © Silent Root".

6. Testing Results

Test Case A: Connectivity & Performance

- **Objective:** Verify system speed and uptime.
- **Result:** The "Fast Scan" mode completed analysis of google.com in **3.5 seconds**. The "Full PDF Report" generation took **12 seconds**.
- **Status: PASSED**.

Test Case B: Accuracy of Tech Detection

- **Objective:** Identify CMS of a known WordPress site.

- **Input:** A test WordPress blog.
- **Result:** System detected wp-content and identified "WordPress" correctly.
- **Status: PASSED**.

Test Case C: Risk Assessment Logic

- **Objective:** Verify if missing headers increase risk score.
- **Input:** HTTP site without SSL and Security Headers.
- **Result:**
  - Detected missing Strict-Transport-Security.
  - Detected missing X-Frame-Options.
  - **Final Score:** 85 (Critical).
- **Status: PASSED**.

Our Team

Ebrahim Hassan {Gui - Algorithm}

Mohamed Tharwat {Flask - request}

Rama Ahmed {Gui - Algorithm}