A blue logo with a bird

AI-generated content may be incorrect.

**Module Name:** Cyber Security Individual Project

**Module Code:** 7030CEM

**Assignment Title:** Project Brief (proposal)

**Project Supervisor:** Florence Nkosi

**Project Supervisor Email:** [ae5962@coventry.ac.uk](mailto:ae5962@coventry.ac.uk)

**Student Name:** Khushal Divyang Patel

**Student ID:** 15405366

**Assignment Due:** 16/06/2025

Contents

[Project Proposal 3](#_Toc200971501)

[ProblemStatement 3](#_Toc200971502)

[Objectives & Research Questions 3](#_Toc200971503)

[Intended Users and Their Requirements 4](#_Toc200971504)

[Proposed Approach 4](#_Toc200971505)

[Tools & Technologies: 5](#_Toc200971506)

[Primary Research Plan 6](#_Toc200971507)

[Initial Literature Review 7](#_Toc200971508)

[Feasibility 8](#_Toc200971509)

[Process Stages and Deliverables: 9](#_Toc200971510)

[Final Project Outcome 10](#_Toc200971511)

[Bibliography 11](#_Toc200971512)

# Project Proposal

Automated Web Vulnerability Scanner

# ProblemStatement

Web applications are prime targets for cyberattacks, with vulnerabilities frequently exploited for unauthorized access or data theft. Many organizations and developers lack the resources or expertise to effectively identify these flaws. Existing security tools are often expensive, complex, or lack user-friendliness. This project addresses the critical need for accessible, lightweight web security assessment. Developing an Automated Web Vulnerability Scanner provides a simplified, user-friendly, web-based tool to help students, ethical hackers, and developers quickly identify common web application vulnerabilities and exposed assets. By automating key reconnaissance and scanning techniques, this tool will lower the barrier to entry for basic security posture assessment, enabling proactive defence against prevalent web-based threats.

# Objectives & Research Questions

**Project Aim**: To develop a lightweight, web-based penetration testing platform for identifying common web application vulnerabilities and sensitive exposures using automated scanning, including Google Dorking and basic network reconnaissance tools.

**Specific Objectives:**

1. Design and implement a secure and intuitive web interface for user interaction and result display.
2. Develop modules for automated hidden file and directory discovery via brute-forcing and public information.
3. Integrate Google Dork-based search functionality to find publicly accessible sensitive information.
4. Implement basic vulnerability checks for common flaws like Cross-Site Scripting (XSS), SQL Injection (SQLi) signature detection, exposed admin panels, and misconfigured directories.
5. Explore optional integration of network scanning tools (e.g., ping sweeps, basic Nmap port scanning).
6. Ensure clear, actionable results and provide report export functionality.

**Research Questions:**

1. How can Google Dorking be effectively automated within a web scanner to identify publicly exposed sensitive files?
2. What are the most effective, lightweight methods for detecting common web application vulnerabilities (XSS, SQLi, exposed panels) without deep code analysis?
3. What are the technical and security considerations for integrating external network scanning tools (like Nmap) into a web platform?
4. How can the user interface deliver clear, concise, and actionable vulnerability reports for varying expertise levels?

# Intended Users and Their Requirements

The primary beneficiaries are students, ethical hackers (beginner to intermediate), and web developers.

* Students: Need an accessible, practical tool for learning web vulnerabilities.
* Ethical Hackers: Require a fast, lightweight tool for initial reconnaissance and identifying common vulnerabilities.
* Web Developers: Seek an easy-to-use tool for rapid self-assessment of their applications.

The project aims to satisfy these needs by providing:

* Ease of Use: Intuitive web interface.
* Accessibility: Web-based, minimal setup.
* Cost-Effectiveness: Free/deployable solution.
* Actionable Insights: Clear reports on detected risks.
* Foundational Assessment: Efficient initial reconnaissance and common vulnerability identification.

# Proposed Approach

**Methodology:** The project will follow an Agile development methodology with iterative development, testing, and feedback. Phases include:

1. **Requirement Analysis & Design:** Detailed feature specification, UI/UX design, and architectural planning.
2. **Module Development:**
   * **Web Interface:** **HTML, CSS, JavaScript** (React considered if time permits).
   * **Backend Logic:** **Python** for scanning, leveraging libraries for requests, parsing.
   * **Hidden File/Directory Discovery:** Algorithms for common paths, robots.txt.
   * **Google Dorking Module:** Automated query construction, result parsing (adhering to usage policies/rate limits).
   * **Vulnerability Signature Checks:** Database of common XSS/SQLi patterns, admin paths, focused on pattern matching.
   * **Network Scanning Integration (Optional):** Securely calling external tools like **Nmap** from Python (sandboxing, error handling).
3. **Testing:** **Unit, integration, and user acceptance testing (UAT)**.
4. **Documentation:** User manuals, developer documentation, project reports.

# Tools & Technologies:

* **Programming Language:** **Python** (backend logic, scanning, potential web framework like Flask/Django).
* **Web Technologies:** **HTML, CSS, JavaScript** (front-end).
* **External Tools:** **Nmap** (optional network scanning); **Google Search API** or web scraping libraries (e.g., requests, BeautifulSoup) for Google Dorking.
* **Development Environment:** **VSCode, Git/GitHub**.
* **Research Instruments:** Academic papers on web security, penetration testing, **OWASP Top 10**.

**Data:**

* **Target Domains:** User-provided URLs (**explicit permission required**).
* **Vulnerability Signatures:** Curated list of common XSS/SQLi patterns, default credentials, exposed directories.
* **Google Dork Queries:** Library of effective queries for sensitive file discovery.

**Project Deliverables and Final Project Outcome**

**Systems Requirements & Characteristics:** The final Automated Web Vulnerability Scanner will be:

* Web-Based Accessible: Browser-only access.
* User-Friendly Interface: Intuitive UI/UX.
* Modular Architecture: Expandable and maintainable.
* Secure: Developed with security best practices.
* Performant: Lightweight and efficient.
* Reporting: Clear result presentation and export (PDF, CSV).

# Primary Research Plan

This project's primary research involves the development and testing of a software demonstrator (the Automated Web Vulnerability Scanner). This practical approach will answer the research questions.

The plan:

1. **Software Development (Application/Software Demonstrator):**
   * **Module-by-Module Implementation:** Incremental building of each scanning component.
   * **Iterative Design & Implementation:** UI/UX evolution based on internal testing.
   * **API Integration:** Researching and developing Python wrappers for external tools (e.g., Nmap).
2. **Experimental Testing (Generated Data):**
   * **Controlled Environment Testing:** Against known vulnerable web applications (e.g., OWASP Juice Shop in isolated lab).
   * **Simulated Scenarios:** Test cases for hidden file discovery and Google Dorking using dummy domains (ethical guidelines apply).
   * **Performance Metrics:** Collection of scan time, resource usage to assess "lightweight" aspect.
3. **User Experience (UX) Validation:**
   * Informal peer feedback (if feasible) on UI usability and report clarity.

This plan emphasizes practical development and empirical testing to validate the project's aims.

# Initial Literature Review

**Introduction:**

This review surveys recent academic contributions in automated web vulnerability scanning, OSINT via search engines, and common web flaw detection. It contextualizes this project within existing research, identifies challenges and approaches, and highlights gaps addressed by this project's lightweight, web-based design with Google Dorking integration.

**Paper 1: Al-Khateeb, A. A., Al-Zubi, S. T., & Mansour, E. S. (2022). A Comprehensive Review of Web Vulnerability Scanners: Taxonomy, Capabilities, and Limitations. Journal of Information Security and Applications, 64, 103038.**

* **Summary:** Reviews web vulnerability scanners, their detection capabilities (XSS, SQLi, misconfigurations), techniques, and limitations (false-positives, logical flaws, authentication).
* **Critical Analysis:** Highlights complexity and resource intensity of comprehensive scanners. Emphasizes accuracy challenges without significant overhead. Focuses less on user accessibility.
* **Relevance:** Underscores the need for a targeted, lightweight scanner. Validates focusing on "basic" vulnerabilities due to inherent limitations of signature-based detection for complex flaws. Supports rapid assessment over exhaustive, resource-heavy scanning.

**Paper 2: Singh, N., & Sharma, M. (2021). Leveraging Google Dorking for Enhanced OSINT and Cyber Reconnaissance. International Journal of Computer Applications, 183(44), 31-35.**

* **Summary:** Demonstrates Google Dorking's power for OSINT, uncovering sensitive information, exposed directories, and configuration files. Discusses utility in initial penetration testing.
* **Critical Analysis:** Focuses on manual Dorking; lacks deep analysis of automation challenges, ethical considerations, or programmatic result parsing.
* **Relevance:** Directly supports Google Dorking as a core feature. Validates its effectiveness for identifying exposed data. My project automates this manual process, addressing the implicit gap in practical, automated OSINT tools for efficiency.

**Paper 3: Chen, P., Wang, Z., Zhang, J., & Li, R. (2023). Lightweight Web Application Vulnerability Detection based on Deep Learning and Signature Matching. Journal of Network and Computer Applications, 211, 103577.**

* **Summary:** Proposes a lightweight detection approach combining signature matching with deep learning for XSS/SQLi, avoiding full code analysis.
* **Critical Analysis:** While integrating deep learning (beyond my initial scope), its emphasis on "lightweight" detection via efficient signature matching is highly relevant. Shows that fast detection is achievable without heavy resource consumption.
* **Relevance:** Reinforces feasibility of "basic vulnerability checks" using signature-based methods. Provides confidence that focusing on known patterns yields quick results, aligning with my project's lightweight goal.

**Synthesis & Gaps:** Literature shows the need for web vulnerability detection. Al-Khateeb et al. reveal scanner limitations and false positives in complex tools. Singh and Sharma highlight Google Dorking's power but omit automated, user-friendly integration. Chen et al. prove lightweight, effective detection is possible via signature matching.

A clear gap is a simple, accessible, web-based tool combining efficient signature-based web scanning with automated Google Dorking for public exposure, specifically for beginner to intermediate users. Existing solutions are either overly complex commercial tools or un-interfaced command-line utilities. My project directly addresses this by consolidating these techniques into an intuitive platform, simplifying initial web security assessment for a broader audience.

**Conclusion:** The review affirms the relevance of web vulnerability scanning and OSINT. There's a demand for lightweight, user-friendly solutions that democratize basic security assessments. This project will contribute by developing an Automated Web Vulnerability Scanner integrating key reconnaissance and detection methods into an accessible web interface, filling a crucial niche for rapid, foundational web security testing.

# Feasibility

**Feasibility:** The project remains highly feasible within the condensed timeline, focusing on "lightweight" and "basic" vulnerability scanning for a functional proof-of-concept. Prioritizing impactful features and deferring complex integrations ensures value delivery. Python's rich library ecosystem is crucial for rapid development. The "optional" nature of network scanning (e.g., Nmap integration) provides flexibility, allowing core functionalities to be completed even under time constraints.

**Potential Challenges & Mitigation:**

* **Aggressive Timeline:** Strict scope adherence, strong time management, focused core feature implementation. Complex integrations may be minimized or deferred.
* **Google Dorking Rate Limits:** Intelligent caching, strategic delays, reliance on pre-curated dork lists.
* **False Positives/Negatives:** Clear caveats for results, emphasizing initial assessment over comprehensive audit.
* **Legal/Ethical Considerations:** User permission is paramount; disclaimers and misuse prevention integral.
* **Security of the Scanner Itself:** Non-negotiable secure coding and input validation from day one.

# Process Stages and Deliverables:

* **Weeks 1-2: Research, Planning & Core Setup**
  + **Deliverables:** Refined Project Plan, System Architecture Document, Initial UI Wireframes, Dev Environment Setup.
* **Weeks 3-5: Core Module Development**
  + **Deliverables:** Functional Python Backends for Hidden File/Directory Discovery and Google Dorking, Initial Vulnerability Signature Library, Backend API Endpoints.
* **Weeks 6-7: Web Interface & Integration**
  + **Deliverables:** Basic Web Front-End (input, scan trigger, basic results), Integrated System Prototype, Preliminary Report Generation.
* **Weeks 8-9: Testing, Refinement & Documentation**
  + **Deliverables:** Comprehensive Test Plan/Results, Bug Fixes/Performance Optimizations, Finalized Core Functionalities, Project Report Draft, User Manual.
  + Optional: Basic Nmap Integration Proof-of-Concept (if time permits).

Final Project Outcome

**Vulnerability Scanner (proof-of-concept)**. This tool will demonstrate the identification of common web application vulnerabilities (XSS/SQLi signatures, exposed files/directories, misconfigured admin panels) and sensitive public exposures via Google Dorking. It directly addresses the problem by providing an accessible, lightweight, and user-friendly solution for basic web security assessments.

# Bibliography

1. Al-Khateeb, A. A., Al-Zubi, S. T., & Mansour, E. S. (2022). A Comprehensive Review of Web Vulnerability Scanners: Taxonomy, Capabilities, and Limitations. Journal of Information Security and Applications, 64, 103038. <https://doi.org/10.1016/j.jisa.2022.103038>
2. Singh, N., & Sharma, M. (2021). Leveraging Google Dorking for Enhanced OSINT and Cyber Reconnaissance. International Journal of Computer Applications, 183(44), 31-35. [You may need to find a direct DOI or stable URL if available from your university's library access. If not, note that it was accessed via specific database and date.]
3. Chen, P., Wang, Z., Zhang, J., & Li, R. (2023). Lightweight Web Application Vulnerability Detection based on Deep Learning and Signature Matching. Journal of Network and Computer Applications, 211, 103577. <https://doi.org/10.1016/j.jnca.2022.103577>