Comprehensive Topic Analysis and Final Selection

Student Name: Abbie Ventour

Course: CSN 190 ASYNC Cybersecurity Project

Instructor: Edwin Reed-Sanchez

Date: Oct. 12th, 2025

Part A: Comprehensive Topic Analysis

Topic 1: Automated Vulnerability Assessment Tool Development (Project-Based)

1. Feasibility Assessment

Technical Requirements:

- Skills/Tools Needed: Python, Bandit, Safety, pip-audit, CVE database integration, JSON parsing, report generation.
- Current Capability (1-5): 3.5 – intermediate Python skills, some security knowledge.
- Need to Learn: Vulnerability scanning frameworks, CVE lookup APIs, and report automation.
- Resources Available: Open-source libraries, public CVE feeds, and test environments (OWASP Juice Shop, DVWA).

Project Scope:

- Build/Test/Deploy: A Python tool that scans applications for vulnerabilities and produces automated reports.
- Environment/Tools: Python 3, VS Code, GitHub, Bandit, Safety, pip-audit.

- Completion Feasibility (8–10 weeks): Achievable with modular development (core scanning and reporting).

2. Learning and Growth Potential

- New Skills: Secure automation scripting, vulnerability detection, API integration, and data reporting.
- Challenges: Parsing vulnerability data and ensuring accurate results.
- Career Alignment: Strongly supports cybersecurity engineering and penetration testing goals.
- Excitement Factor: Building a working tool that automates real security tasks.
- Sustainability: High – hands-on coding and practical results will sustain motivation.

3. Realistic Planning

- Ambitious Goal: Full-featured tool that scans multiple targets with detailed HTML reporting.
- Realistic Target: Working Python tool that detects vulnerabilities from CVE data and outputs text/JSON reports.
- Minimum Viable Outcome: Basic scanner that runs vulnerability checks using Bandit and Safety.

Topic 2: Secure Code Review – Common Vulnerabilities in Python Applications (Project-Based)

1. Feasibility Assessment

Technical Requirements:

- Skills/Tools Needed: Python, Bandit, Flake8, pylint, CWE database familiarity, static analysis techniques.
- Current Capability (1-5): 3 – solid Python but limited experience with static code analysis.
- Need to Learn: Writing custom static analysis rules and mapping vulnerabilities to CWE identifiers.

- Resources Available: Open-source static analysis tools, sample insecure Python projects, CWE database.

Project Scope:

- Build/Test/Deploy: A tool that scans Python source code for security issues and flags them.
- Environment/Tools: Python 3, VS Code, Bandit, Flake8, CWE dataset.
- Completion Feasibility (8–10 weeks): Moderate – custom rule creation could extend timeline.

2. Learning and Growth Potential

- New Skills: Secure code analysis, CWE classification, Python AST (Abstract Syntax Tree) analysis.
- Challenges: Understanding complex static analysis logic.
- Career Alignment: Ideal for future work in DevSecOps or application security.
- Excitement Factor: Investigating how small code errors can lead to big vulnerabilities.
- Sustainability: Medium-high – research and analysis-heavy, less hands-on than Topic 1.

3. Realistic Planning

- Ambitious Goal: A custom code analysis tool that scans Python code for multiple CWE vulnerabilities.
- Realistic Target: A tool that uses Bandit to detect and classify common Python vulnerabilities.
- Minimum Viable Outcome: A report generated from Bandit analysis mapped to CWE categories.

Part B: Comparison Table

Criteria	**Topic 1: Automated Vulnerability Assessment Tool**	**Topic 2: Secure Code Review Tool**
Feasibility/Accessibility	**5**	**4**
Personal Interest	**5**	**4**
Learning Potential	**5**	**5**
Career Relevance	**5**	**5**
Total	**20/20**	**18/20**

Part C: Final Topic Selection and Justification

Final Choice:

Automated Vulnerability Assessment Tool Development (Project-Based)

Justification:

After analyzing both potential research directions, I have chosen to pursue the Automated Vulnerability Assessment Tool Development project. This project provides an ideal balance between technical challenge, career relevance, and personal engagement. The goal is to create a Python-based tool that automates the detection of security weaknesses, outdated dependencies, and configuration issues in applications or small networks.

This topic is achievable yet intellectually stimulating. It builds on my current Python experience and introduces advanced cybersecurity concepts like CVE integration, vulnerability categorization, and report automation. Unlike many academic research topics, this project will result in a tangible, functioning product—one that mirrors the kind of automation tools used by professional penetration testers and security analysts. It is challenging enough to expand my understanding of vulnerability scanning but feasible within the 8–10 week timeframe due to the accessibility of open-source libraries like Bandit and Safety.

This project is personally meaningful because it contributes directly to proactive security practices—identifying vulnerabilities before exploitation occurs. It aligns perfectly with my long-term goal of working in cybersecurity engineering, where automation and efficiency are increasingly valued.

My three-tier plan includes:

- Ambitious Goal: Develop a robust, multi-platform scanning tool with a user interface and detailed HTML reports.

- Realistic Target: Deliver a working Python script capable of scanning for common vulnerabilities and generating structured text or JSON reports.

- Minimum Viable Outcome: Implement a command-line tool that successfully identifies vulnerabilities using Bandit and Safety.

The learning value lies not only in coding the tool but in understanding the vulnerability assessment process end-to-end—from identifying risks to presenting results. Even if the full feature set is not achieved, this project ensures growth in secure coding, automation, and Python security scripting. Attempting this project demonstrates initiative, technical depth, and readiness for real-world cybersecurity roles.

(Word count: 362)