

Project Proposal

04 February 2026 09:58

Project Proposal

Module: Changemakers: Creativity and Value
Creation (ACCA5033)

Student Name: Skyler Schenke

Student ID: 2405518

Programme: BSc Cyber Security & Computer
Networks

Assignment: Innovation & Project Management
Proposal

Project Title:

Automated Security Misconfiguration Detection
Tool for Configuration Baseline Analysis

Lecturer: James MG Williams

Submission Date: 10 March 2026

1.1 Innovation and the Role of Creativity in Cybersecurity

04 February 2026 09:57

1.1 Innovation and the Role of Creativity in Cybersecurity

Innovation refers to the development or improvement of ideas, technologies, or processes that create value within society. While innovation is often associated with entirely new inventions, many innovations involve incremental improvements that increase efficiency, reliability, and effectiveness in existing systems. Creativity is a key driver of innovation because it enables individuals to recognise problems, explore alternative perspectives, and design solutions that improve real-world technological systems [9]. In modern digital environments, innovation is particularly important in cybersecurity because both technologies and threats evolve rapidly.

Organisations increasingly rely on complex digital infrastructure including cloud platforms, distributed networks, and interconnected software systems. As these systems grow in scale and complexity, maintaining their security becomes increasingly difficult. System administrators must manage numerous configuration settings, permissions, and service policies. Incorrect configurations can introduce vulnerabilities that attackers may exploit, making configuration management a critical aspect of modern cybersecurity practice [2].

Security misconfiguration is widely recognised as one of the most common causes of system vulnerabilities. The Open Web Application Security Project (OWASP) identifies misconfiguration as a major security risk because insecure system settings can expose sensitive information, weaken authentication mechanisms, or allow unauthorised system access [1]. These issues frequently occur due to the complexity of modern infrastructures and the large number of configuration parameters administrators must manage.

Automation therefore represents an important area of innovation within cybersecurity. Automated security tools can analyse system configurations, compare them against recognised security baselines, and detect weaknesses before they are exploited [6]. However, automation must be implemented carefully because cybersecurity decisions often require contextual understanding. Automated systems should therefore support human judgement rather than replace it entirely.

Cybersecurity innovation also has wider societal value. Secure digital systems underpin essential services such as healthcare infrastructure, financial systems, communication networks, and government platforms. Improving the ability of organisations to detect security weaknesses strengthens digital resilience and increases trust in technology [7]. Consequently, tools that assist administrators in identifying configuration vulnerabilities represent a valuable opportunity for innovation within cybersecurity.

1.2 Ideation and Project Idea Selection

04 February 2026 10:00

1.2 Ideation and Project Idea Selection

During the ideation stage of this project, several cybersecurity-related project ideas were explored. Each idea was evaluated based on its relevance to the programme of study, feasibility within a student project, and potential impact on improving cybersecurity practices.

The first idea involved developing a phishing risk analysis tool capable of analysing phishing email characteristics and estimating the likelihood that users might interact with malicious messages. Although phishing attacks represent a significant cybersecurity threat, developing accurate behavioural analysis would require extensive datasets and user testing, making it difficult to implement within a limited prototype project.

A second idea involved creating an authentication configuration simulator designed to demonstrate how security policies such as password complexity requirements, multi-factor authentication, and account lockout rules influence system security. While this idea would provide educational value, it would primarily function as a simulation tool rather than solving a real operational cybersecurity problem.

A third concept involved building a privacy-focused configuration checker capable of evaluating system settings against data protection guidelines. However, implementing a comprehensive compliance tool would require significant legal and regulatory expertise, which exceeds the realistic scope of this project.

After evaluating these alternatives, the selected idea is the development of an automated security misconfiguration detection prototype. This project was chosen because configuration errors frequently contribute to security breaches, meaning that tools capable of detecting insecure system settings can significantly improve organisational security practices [8]. Additionally, the concept remains technically achievable within a controlled prototype environment.

Project Title

Automated Security Misconfiguration Detection Tool for Configuration Baseline Analysis

Project Aim

To design and prototype a rule-based system capable of detecting security misconfigurations by comparing configuration settings against recognised security baselines.

Project Objectives

- Identify common configuration weaknesses within modern computing environments.
- Develop rule-based detection logic derived from recognised cybersecurity standards.
- Implement a prototype capable of analysing configuration data.
- Generate structured reports that classify detected risks according to severity.

- Evaluate the limitations and ethical considerations associated with automated configuration analysis tools.

1.3 Research Sources Supporting the Project

04 February 2026 10:00

1.3 Research Sources Supporting the Project

The development of this project is supported by established cybersecurity frameworks and research sources. Security configuration benchmarks published by the Center for Internet Security provide structured guidance for securely configuring operating systems and network services [4], [5]. These benchmarks are widely adopted by organisations to establish consistent security baselines.

Guidance from the National Institute of Standards and Technology further emphasises the importance of configuration management and baseline security controls [2], [3]. NIST publications highlight how maintaining secure configurations and monitoring systems for unauthorised changes can significantly reduce vulnerabilities.

Existing security auditing tools such as OpenSCAP and Lynis demonstrate current approaches to automated configuration analysis [10], [11]. While these tools provide useful auditing capabilities, they often prioritise compliance reporting rather than decision-support functionality. The proposed project therefore explores a simplified rule-based detection approach that emphasises transparency, explainability, and educational value within a prototype environment.

2.1 Project Scope and Proposed Solution

04 February 2026 09:57

2.1 Project Scope and Proposed Solution

This project aims to design and develop a prototype automated security misconfiguration detection tool. The system analyses configuration data, compares system settings against recognised security baselines, and identifies potential configuration vulnerabilities. The tool is designed to support cybersecurity decision-making rather than fully automate security management.

The proposed solution uses a rule-based evaluation model. Configuration values are compared against baseline rules derived from recognised cybersecurity standards such as those published by the Center for Internet Security and guidance from the National Institute of Standards and Technology [2], [4]. When configuration settings deviate from recommended secure values, the system records the issue as a potential misconfiguration and assigns an appropriate severity level.

The analysis workflow begins with a configuration file representing system settings. The system parses this data to extract relevant parameters such as authentication policies, service states, and permission settings. These parameters are then evaluated by a rule engine that compares them against predefined security baselines. If deviations are detected, the system records the finding and generates a structured security report describing the issue and its potential impact.

The prototype is implemented using Python because the language provides strong support for structured data processing and rapid development of rule-based logic [6]. Python also supports modular development, which improves maintainability and debugging. Version control is managed using GitHub in order to track development changes.

Testing is conducted using simulated configuration datasets rather than real systems. Example configuration files containing both secure and insecure values allow the rule engine to detect misconfigurations and validate the reporting functionality. This controlled testing environment ensures that the project remains safe and ethical.

The project deliberately excludes advanced capabilities such as real-time network scanning, automated remediation, or enterprise deployment. These features would significantly increase complexity and fall outside the scope of a student prototype. Instead, the project focuses on demonstrating how rule-based configuration analysis can support cybersecurity decision-making.

2.2 Project Timeline and Gantt Chart

04 February 2026 10:01

2.2 Project Timeline and Gantt Chart

To ensure the project is completed within the available timeframe, development activities are organised into structured phases. Each stage builds upon the previous stage and allows the system to be developed systematically.

The first phase focuses on research and planning. During this stage, relevant cybersecurity frameworks and configuration benchmarks are analysed to identify baseline rules for the detection system [4], [5].

The second phase focuses on system design. During this stage the architecture of the prototype is defined, including configuration input mechanisms, rule evaluation logic, severity classification models, and reporting components.

The third phase involves implementing the prototype. Configuration parsing and rule evaluation mechanisms are implemented using Python, allowing the system to analyse sample configuration datasets.

The fourth phase involves testing and validation. Simulated configuration datasets are used to evaluate detection accuracy and confirm that severity classification operates correctly.

The final phase involves documentation and preparation of the project submission. This stage includes organising design artefacts, documenting system functionality, and preparing the final project proposal.

A Gantt chart is used to visually represent these phases and illustrate the sequence of development tasks across the project timeline.

Figure 1 – Project Development Timeline (Gantt Chart)

Project Development Timeline (Gantt Chart)



Figure 1: Project development timeline showing the planned stages of research, system design, implementation, testing, and documentation for the automated misconfiguration detection prototype.

2.3 Risk Analysis and Mitigation Strategies

04 February 2026 10:01

2.3 Risk Analysis and Mitigation Strategies

Several risks may affect the development of the prototype. Identifying these risks early allows appropriate mitigation strategies to be implemented.

One potential risk is the generation of false positives. Because rule-based systems rely on predefined baseline rules, legitimate configurations may occasionally be flagged as insecure. Excessive false positives could reduce user trust in the system. To mitigate this issue, the prototype includes severity classification and explanatory reporting to provide context for each detected issue.

Another risk involves incorrect implementation of baseline rules. Errors in rule logic could lead to inaccurate detection results. This risk will be mitigated through testing with controlled datasets containing both secure and insecure configuration examples.

Scope creep represents another potential challenge. Adding unnecessary features may increase project complexity and delay development. The project scope has therefore been carefully defined to focus on building a functional prototype.

Finally, software instability may occur during development. Modular testing will be used to verify individual components before integrating them into the complete system.

3.1 Development Methodology

04 February 2026 09:58

3.1 Development Methodology

An iterative development methodology is used for this project. Iterative development builds systems through successive stages, where each iteration improves the previous version. This approach is particularly suitable for cybersecurity tools because detection logic often requires testing and refinement [6].

The first iteration defines the project scope and identifies relevant configuration baseline rules based on recognised cybersecurity frameworks such as NIST guidance and CIS benchmarks [2], [4]. The second iteration focuses on designing the system architecture, including configuration input mechanisms, rule evaluation logic, and reporting structures.

The third iteration implements the rule engine and configuration analysis functionality. The fourth iteration focuses on testing and refinement using simulated configuration datasets. The final iteration improves usability, reporting output, and documentation.

This iterative approach allows issues such as incorrect rule behaviour or excessive false positives to be identified and resolved early in the development process.

3.2 Design Work

04 February 2026 10:02

3.2 Design Work

Two design artefacts support the development of the prototype: a system architecture diagram and a rule engine analysis flow diagram.

Figure 2: System architecture of the automated security misconfiguration detection system.

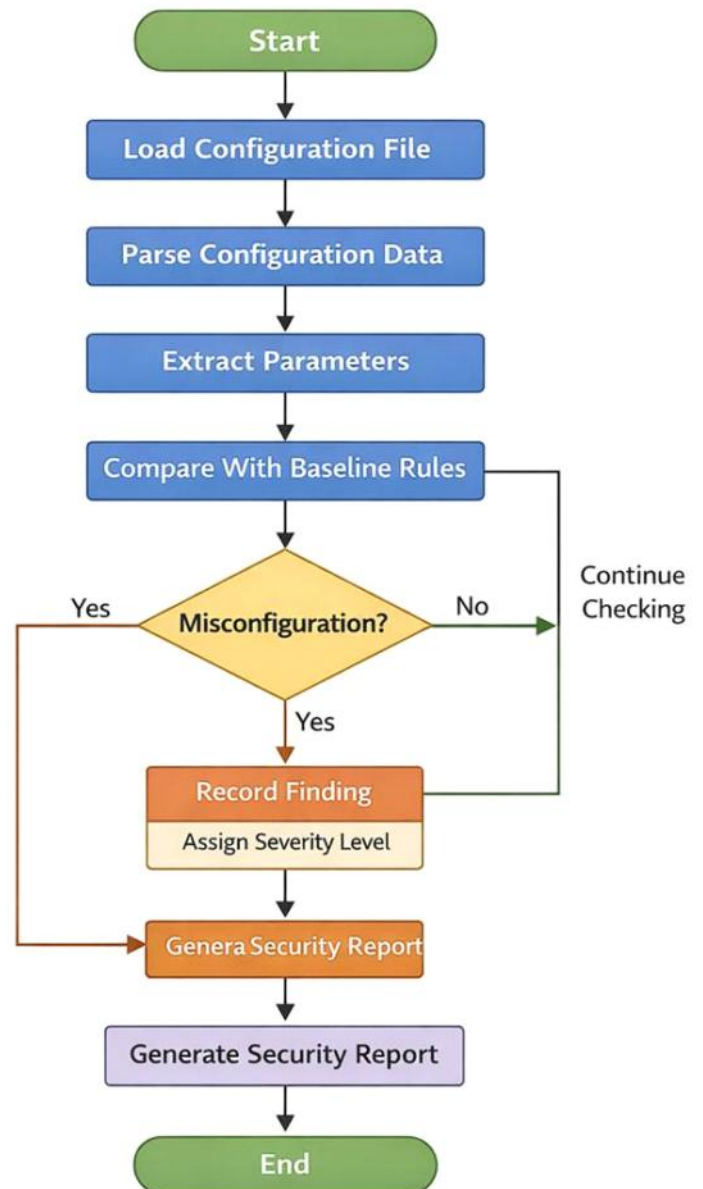


The system architecture diagram illustrates how configuration data flows through the system. The configuration input module receives system settings from a configuration file. The configuration parser extracts relevant parameters and converts raw configuration data into a structured

format suitable for analysis.

The rule engine evaluates these parameters against predefined baseline rules derived from recognised cybersecurity standards such as CIS benchmarks and NIST configuration management guidance [2], [5]. If a configuration value deviates from the secure baseline, the system records a misconfiguration finding. The severity classification module then assigns a risk level based on the potential security impact, and the reporting module generates a structured report describing the detected vulnerabilities.

Figure 3: Rule engine workflow used to evaluate configuration settings against security baselines.



The rule engine flow diagram illustrates the logical sequence of operations used during configuration analysis. Configuration data is loaded and parsed, relevant parameters are extracted, and each value is compared against predefined baseline rules. When a deviation is detected, the system records the misconfiguration, assigns a severity level, and includes the result in the generated security report.

3.3 Initial Practical Implementation and Evidence

04 February 2026 10:02

3.3 Initial Practical Implementation and Evidence

Initial implementation focuses on developing the rule-based configuration analysis mechanism using Python.

Figure 4: Python project structure used to implement the prototype system.

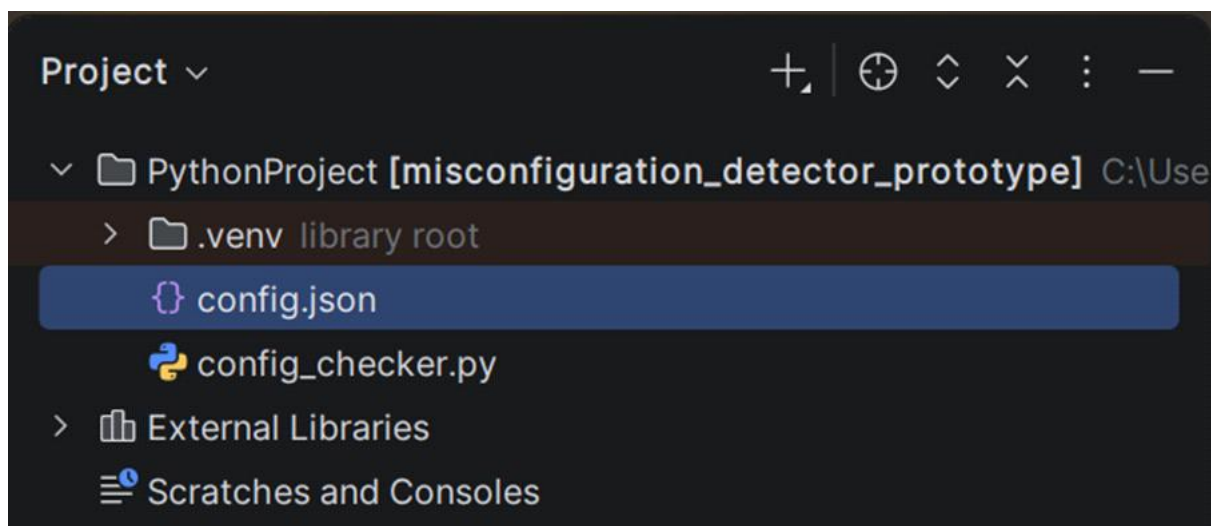


Figure 4 shows the development environment used to build the prototype system. The project directory includes a configuration file (config.json) containing system settings and a Python script (config_checker.py) responsible for analysing configuration values and detecting potential misconfigurations. A Python virtual environment is used to manage project dependencies.

Figure 5: Example configuration dataset used for prototype testing.

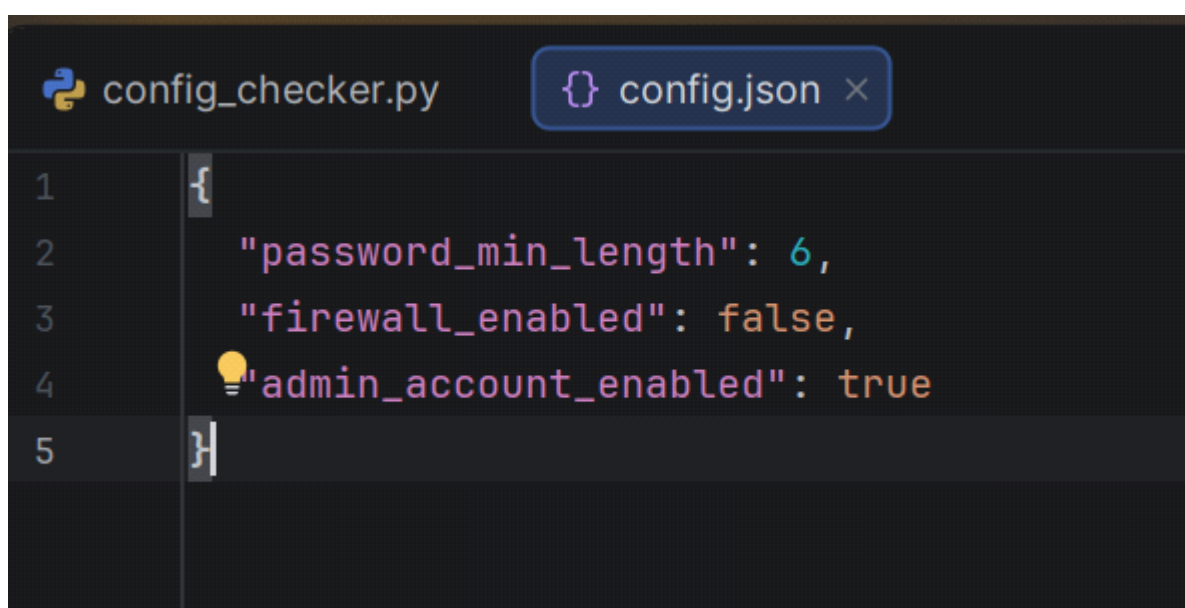
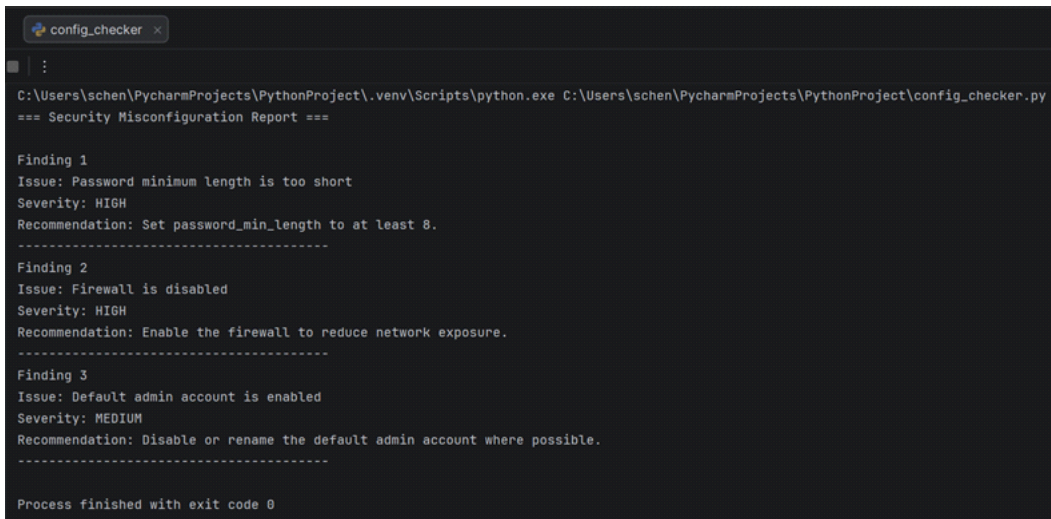


Figure 5 shows a sample configuration file used to test the misconfiguration detection prototype. The configuration dataset contains both secure and intentionally insecure settings representing system parameters such as authentication

configuration and service permissions. These test values allow the rule engine to detect configuration weaknesses and evaluate the accuracy of the detection logic.

Figure 6: Example output generated by the rule-based misconfiguration detection system.



```
config_checker x
C:\Users\schen\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\schen\PycharmProjects\PythonProject\config_checker.py
=== Security Misconfiguration Report ===

Finding 1
Issue: Password minimum length is too short
Severity: HIGH
Recommendation: Set password_min_length to at least 8.
-----
Finding 2
Issue: Firewall is disabled
Severity: HIGH
Recommendation: Enable the firewall to reduce network exposure.
-----
Finding 3
Issue: Default admin account is enabled
Severity: MEDIUM
Recommendation: Disable or rename the default admin account where possible.
-----

Process finished with exit code 0
```

Figure 6 demonstrates the output produced by the prototype after analysing the configuration dataset. The system identifies configuration settings that deviate from recognised security baselines and assigns a severity classification based on the potential impact of the issue. The structured output enables administrators to quickly identify and prioritise configuration vulnerabilities.

These screenshots demonstrate that the prototype system has been partially implemented and provide practical evidence supporting the feasibility of the proposed approach.

References

04 February 2026 10:03

References

- [1] OWASP Foundation, “OWASP Top 10:2021 – The Ten Most Critical Web Application Security Risks,” 2021. Available: <https://owasp.org/www-project-top-ten/>
- [2] National Institute of Standards and Technology, Security and Privacy Controls for Information Systems and Organizations, NIST Special Publication 800-53 Rev. 5, 2020. Available: <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>
- [3] National Institute of Standards and Technology, Guide for Security-Focused Configuration Management of Information Systems, NIST Special Publication 800-128, 2011. Available: <https://csrc.nist.gov/publications/detail/sp/800-128/final>
- [4] Center for Internet Security, CIS Critical Security Controls v8, 2021. Available: <https://www.cisecurity.org/controls/v8>
- [5] Center for Internet Security, CIS Benchmarks for Secure System Configuration, 2023. Available: <https://www.cisecurity.org/cis-benchmarks>
- [6] E. Matthes, Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2nd ed. San Francisco, CA: No Starch Press, 2019.
- [7] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 3rd ed. Indianapolis, IN: Wiley, 2020.
- [8] Verizon, 2024 Data Breach Investigations Report, 2024. Available: <https://www.verizon.com/business/resources/reports/dbir/>
- [9] H. Chesbrough, Open Innovation: The New Imperative for Creating and Profiting from Technology. Boston, MA: Harvard Business School Press, 2003.
- [10] OpenSCAP Project, OpenSCAP Security Compliance Solution, 2023. Available: <https://www.open-scap.org/>
- [11] CISOfy, Lynis Security Auditing Tool, 2023. Available: <https://cisofy.com/lynis/>