The British College

Cyber Security and Digital Forensics

Cyber Security Project

Project Proposal Document

Module Tutor: Santosh Sharma

Student Name: Isha Bhetwal

UWE ID: 23045173

MCPSecurityTool: An AI-Based Prompt Injection and Misconfiguration Security Analysis Tool

Isha Bhetwal

20/11/2025

Contents

1 Introduction

The increasing adoption of AI systems has simultaneously increased exposure to adversarial risks such as prompt injection, model exploitation, and configuration weaknesses (OpenAI 2024). Large Language Models (LLMs) are now integrated into critical infrastructures, making their security a priority (Zhang et al. 2023). The Model Context Protocol (MCP) provides a structured framework for tools interacting with LLMs, but it is highly vulnerable when misconfigured (Raff 2022). Therefore, securing these systems demands specialized automated testing methods (Carlini et al. 2024). The MCPSecurityTool is proposed as a machine-learning-driven solution to identify weaknesses in MCP implementations (Wei et al. 2022). This tool bridges the emerging gap between rapid AI adoption and insufficient AI-security expertise (Pressman & Maxim 2020).

1.1 Current Scenario in Nepal

Nepal's digital transformation has accelerated since 2020, with banks, telecoms, and public institutions integrating AI-based automation (MoCIT 2023). However, national cybersecurity maturity remains significantly behind global standards (Pandey 2022). Security teams often lack expertise in AI-specific vulnerabilities such as prompt injection (Zhang et al. 2023). Nepal's increasing cyber incidents indicate systemic weaknesses in AI deployment strategies (Pandey 2022). This necessitates a locally accessible, automated AI-security testing tool tailored to Nepal's emerging tech environment (MoCIT 2023).

1.2 Project as a Solution

The MCPSecurityTool aims to automatically detect prompt injection vulnerabilities, misconfigurations, and unsafe execution pathways in MCP-based systems. The tool uses a machine learning classifier to evaluate model behaviour under adversarial prompts. It generates structured security reports for system administrators and developers. By offering automated threat simulation, the tool reduces the reliance on manual security auditing. Thus, it directly addresses Nepal's AI-security gap with a scalable and localisable solution.

2 Expected Outcomes and Deliverables

The project will produce a functional prototype capable of detecting prompt injection threats. Deliverables include a full MCP vulnerability scanner and misconfiguration detection engine. The tool will generate detailed reports summarizing vulnerability types and severity. A dataset of adversarial prompts will also be created for future research. Additional deliverables include documentation, deployment guides, and demonstration.

3 Project Risks and Contingency Plans

Potential risks include dataset limitations due to scarce AI-security data (Wei et al. 2022). Mitigation includes generating synthetic adversarial prompts (Carlini et al. 2024). Model overfitting is another risk, mitigated by cross-validation and periodic retraining (Pressman & Maxim 2020). MCP tool behaviour may vary across LLM implementations, requiring multi-model testing (OpenAI 2024). Fallback rule-based detection will be implemented in case ML models fail (Raff 2022).

4 Methodology

4.1 Selected Methodology

A hybrid methodology combining research, engineering, and iterative development is selected. This ensures that theoretical understanding aligns with practical implementation.

4.2 Agile Methodology (SCRUM Framework)

SCRUM enables rapid prototyping, frequent testing, and continuous improvement (Schwaber & Sutherland 2020). Sprints will be one to two weeks long with defined sprint backlogs. Daily standups ensure progress tracking and coordination (Schwaber & Sutherland 2020). Sprint reviews demonstrate functional increments of the MCPSecurityTool (Pressman & Maxim 2020).

5 Resource Requirements

5.1 Hardware Requirements

A workstation with at least an Intel i5 processor or equivalent is required for model development. 16GB RAM is recommended for data processing and training tasks. A dedicated GPU accelerates deep-learning tasks.

5.2 Software Requirements

Python 3.10+ and ML libraries such as TensorFlow or PyTorch will be used. Scikit-learn is required for classical machine learning algorithms. The MCP SDK is essential for interacting with MCP-compatible systems. GitHub will be used for version control and collaboration.

6 Work Breakdown Structure

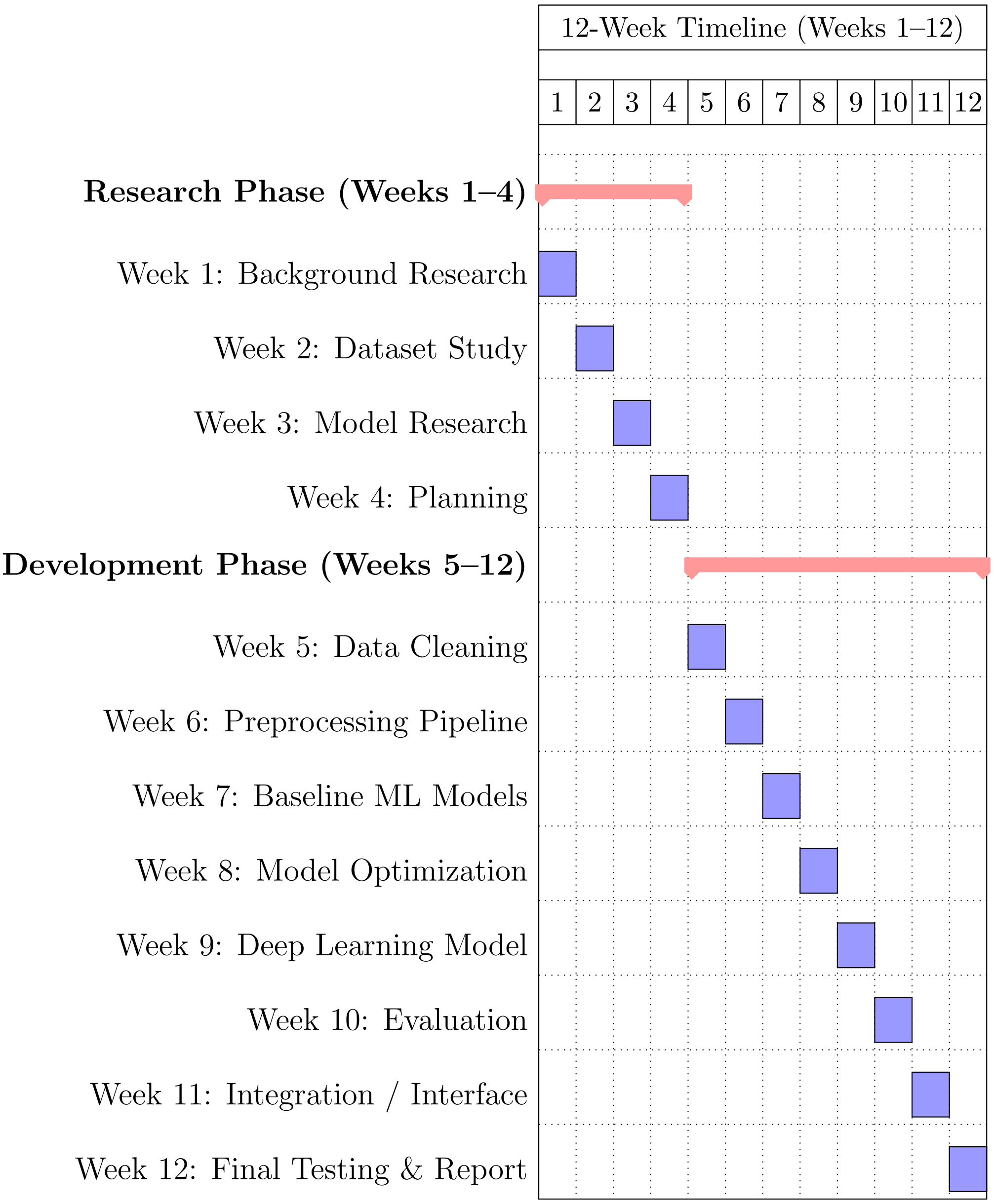
The WBS includes research, dataset preparation, model development, tool integration, evaluation, and documentation. Research tasks include background study and threat modelling. Development tasks include preprocessing pipelines and ML model creation. Evaluation includes testing adversarial prompts across multiple LLMs

7 Milestones

Major milestones include completion of research by Week 4.
ML model completion by Week 10.
Interface and integration completed by Week 11.
Final report completion by Week 12.

8 Gantt Chart

Project Duration: 2 October 2025 – 3 January 2026



9 Conclusion

The MCPSecurityTool will significantly contribute to AI-security research by providing automated detection of prompt injection and MCP misconfiguration vulnerabilities (Wei et al. 2022). It will support Nepal's digital ecosystem by offering accessible security evaluation tools (Pandey 2022). The project's combination of machine learning and systematic testing provides a scalable long-term solution (Raff 2022).

10 References

References

Carlini, N., Jagielski, M. & Mironov, I. (2024). Adversarial prompt attacks on large language models. Google DeepMind Research Papers.

MoCIT. (2023). National Cybersecurity Strategy of Nepal. Ministry of Communications and Information Technology.

OpenAI. (2024). Model Context Protocol (MCP) Technical Overview. OpenAI Documentation.

Pandey, S. (2022). Cybersecurity readiness in Nepalese enterprises. Journal of Information Security, 9(2), 44–55.

Pressman, R. & Maxim, B. (2020). Software Engineering: A Practitioner's Approach. McGraw-Hill.

Raff, E. (2022). Inside Machine Learning Security. MIT Press.

Schwaber, K. & Sutherland, J. (2020). The Scrum Guide. Scrum.org.

Wei, J., Wang, X. & Zhou, D. (2022). Jailbreak vulnerabilities in LLM systems. arXiv preprint arXiv:2211.09110.

Zhang, Y., Li, S. & Chen, X. (2023). Security risks in LLM-integrated systems. IEEE Security & Privacy, 21(1), 55–63.