A Zero Trust Architecture for the Model Context Protocol

by

Abhishek Bhise

Dissertation submitted in partial fulfilment for the Degree of Master of Science
in
Cyber Security Management

WMG
University of Warwick

Submitted Sept. 2025

Declaration

I have read and understood the rules on cheating, plagiarism and appropriate referencing as outlined in my handbook and I declare that the work contained in this assignment is my own, unless otherwise acknowledged.

No substantial part of the work submitted here has also been submitted by me in other assessments for this or previous degree courses, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

Project definition for my degree (as copied from the Specific Course Regulations for your course, available here: https://warwick.ac.uk/fac/sci/wmg/ftmsc/opregs/courseregs/)

The dissertation for MSc Cyber Security Management must:

- demonstrate managerial/consultation/leadership insight and depth, and can involve surveys AND
- address a research question directly relating to one or more CyBOK Knowledge Areas, AND
- demonstrate understanding of the particular issues around conducting research in the cyber domain AND
- conduct research in the cyber domain in an appropriate manner.

My project relates to this definition in the following way:

This dissertation provides significant "managerial and consultation insight" by addressing the "research question" of how to architect a strategic Zero Trust solution for agentic AI. The work directly relates to key "CyBOK Knowledge Areas," including Security Architecture and Risk Management. It "demonstrates a deep understanding of the particular issues around conducting research in the cyber domain" by analysing novel AI-native threats like "Tool Poisoning" and "Rug Pull" attacks. Finally, the project was completed by "conducting research in the cyber domain in an appropriate manner," using a systematic methodology that synthesizes authoritative frameworks like ZTA and SABSA to construct a robust and enterprise-ready security architecture.

Abstract

The rapid adoption of agentic AI, enabled by standards like the Model Context Protocol (MCP), is creating a new paradigm of interoperable, autonomous systems. However, this advancement introduces a significant and poorly understood attack surface, as MCP's open, client-server architecture exposes agentic systems to novel threats such as "Tool Poisoning," "Puppet Attacks," and "Rug Pull Attacks" that can lead to data exfiltration and unauthorised actions. While Zero Trust Architecture (ZTA) provides the necessary security philosophy of "never trust, always verify," a critical gap exists between its high-level principles and a practical, systematic method for applying them to the unique risks of the MCP ecosystem.

This dissertation employs a quantitative modelling design to address this gap. The methodology involves a systematic analysis of the MCP threat landscape, drawing on publicly available research, industry reports, and open-source deployments. It then uses the business-driven, multi-layered SABSA framework as an analytical tool to architect a comprehensive ZTA specifically tailored to secure agentic AI communications over MCP.

The key findings demonstrate that the identified threats exploit the protocol's implicit trust model and can be effectively mitigated through a structured, layered security architecture. The research culminates in the design of a ZTA for MCP where security controls are directly traceable to business requirements and risks. This architecture uses ZTA's logical components such as the Policy Engine, Policy Administrator, and Policy Enforcement Point to enforce granular, context-aware security at critical interaction points.

The key contribution of this work is a practical, business-driven, and traceable security blueprint for the secure adoption and governance of agentic AI. It demonstrates how established enterprise architecture methodologies can be effectively applied to mitigate the new threats posed by emerging technological paradigms, transforming MCP from a high-risk protocol into a secure enabler of AI innovation.

Acknowledgements

As Isaac Newton once said, “If I have seen further, it is by standing on the shoulders of giants.” I find it a sincere challenge to fit all my gratitude onto a single page, but I must try. This master's study has been the most extraordinary chapter of my life, and who I am today and the potential I see in my future endeavours would not have been possible without some incredible people. I would like to take this opportunity to express my deepest gratitude to these giants who were part of this journey with me.

First and foremost, I could never be the man I am without my family. I count myself among the luckiest of people who had the privilege to be raised in such a supportive and nourishing home. I offer my deepest gratitude to my mother, whose endless love and care have been a constant source of strength and comfort. To my father, for his never-ending support and for always being there for me, no matter the challenge. To my brother, for being a great sport, a source of laughter, and for always having my back. Your collective belief in me has been my greatest motivation, and for that, I am eternally grateful.

This work owes its existence to guidance and wisdom of my mentors. I am especially indebted to my dissertation supervisor. I have never met a person so kind, considerate, and genuinely invested in his students' success. His guidance was a guiding beacon during times of uncertainty, and his door was always open. I cannot thank him enough, and this dissertation is as much a testament to his mentorship as it is to my effort. I would also like to extend my sincere thanks to the other professors in Cybersecurity Management (MSc) program for their invaluable lessons and support. The lessons and knowledge would stay with me for life. My gratitude also goes to all the special teachers I have had the privilege to study under throughout my life, who ignited my curiosity and passion for learning.

Finally, I must thank my friends, who have made this journey so much more enjoyable. For the shared laughter, experiences and adventure. Thank you for being a constant source of help and fun during the most stressful of times. Your friendship has been an invaluable gift.

While many have contributed to the success of this project, any errors or omissions that remain are, of course, my own.

Table of Contents

List of tables

List of figures

List of Abbreviations and Key Terms

Term / Abbreviation	Definition / Explanation
AI	Artificial Intelligence.
DIDs	Decentralised Identifiers.
ETDI	Enhanced Tool Definition Interface. A security extension for MCP designed to counter "Tool Poisoning" and "Rug Pull" attacks through cryptographic identity, immutable definitions, and explicit permissions.
JIT	Just-in-Time. A security principle related to providing credentials or access for the minimum time necessary to perform a task.
JWS	JSON Web Signature. A standard (IETF RFC 7515) for representing signed content using JSON data structures, used in this architecture to sign tool definitions.
JWT	JSON Web Token. A standard (IETF RFC 7519) for creating access tokens that assert a number of claims, used here for implementing Just-in-Time credentials.
LLM	Large Language Model.
MCP	Model Context Protocol.
MITRE ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge.
MITRE ATLAS	Adversarial Threat Landscape for Artificial-Intelligence Systems
MPMA	Preference Manipulation Attack. A subtle attack where a tool's description is manipulated with persuasive language to make an LLM more likely to select it over competitors, often for economic gain.
mTLS	Mutual Transport Layer Security. An extension of TLS where both the client and server authenticate each other's identity using digital certificates, providing secure service-to-service communication.
NIST	National Institute of Standards and Technology.
OPA	Open Policy Agent. An open-source, general-purpose policy engine that enables unified, context-aware policy enforcement.
PA	Policy Administrator.
PBAC	Policy-Based Access Control.
PDP	Policy Decision Point.
PE	Policy Engine.
PEP	Policy Enforcement Point.
PKI	Public Key Infrastructure.
Puppet Attack	An attack where a malicious tool does not perform a harmful action itself but instead manipulates the LLM to misuse other legitimate, trusted tools to carry out unauthorised actions.

RADE	Retrieval-Agent Deception. An advanced attack where an attacker embeds malicious prompts within external data sources, which are then ingested by a retrieval agent and executed, coercing the LLM into performing harmful actions.
RBAC	Role-Based Access Control. An access control model where permissions are assigned to roles rather than individual users.
Rug Pull	A "bait-and-switch" attack where the functionality of an approved tool is maliciously altered by its provider after the user has granted initial consent.
SABSA	Sherwood Applied Business Security Architecture.
SIEM	Security Information and Event Management.
SOC	Security Operations Centre.
Tool Poisoning	An attack where a malicious actor deploys a tool that masquerades as a legitimate or harmless one, deceiving a user or LLM into selecting and using it for nefarious purposes.
Tool Squatting	The deceptive registration of a tool with a name similar to a trusted tool to mislead discovery mechanisms and trick users or agents into selecting the malicious version.

Chapter 1: Introduction

1.1 Introduction

This chapter sets the stage for the dissertation, introducing the agentic AI landscape, the central security problem, and the overall structure of the investigation. Section 1.2 will explore the security challenges of the Model Context Protocol's (MCP) broken trust model. Section 1.3 will then formally define the research problem, aim, and question, focusing on the lack of a verifiable trust framework. Following this, Section 1.4 articulates the research's significance, while Section 1.5 outlines the dissertation's organisation. Finally, Section 1.6 provides a concise summary of the chapter.

1.2 The Security Challenge in Agentic Ecosystems

The field of Artificial Intelligence is undergoing a profound transformation, moving beyond the capabilities of passive Large Language Models to proactive "agentic" systems that can perceive their environment, formulate plans, and execute actions by interacting with external tools and data sources (H. Song *et al.*, 2025; Kumar *et al.*, 2025). Powering this revolution is the Model Context Protocol (MCP), an open standard that addresses the challenge of fragmented tool integration by providing a universal, client-server protocol that standardises how AI applications discover and use tools (Hasan *et al.*, 2025; Hou *et al.*, 2025). The protocol's potential has been recognised quickly, leading to its widespread adoption by industry leaders and establishing it as a de facto standard for AI-to-tool communication (Google, 2025; Hasan *et al.*, 2025; Microsoft, 2025a).

While the flexibility and openness of MCP are the primary drivers of its success, they are also the source of its most significant security challenges (Bhatt, Narajala and Habler, 2025; Kumar *et al.*, 2025). The current MCP specification operates on a fundamentally broken trust model, where AI applications acting as clients implicitly trust the identity and integrity of MCP servers and the tools they expose (Bhatt, Narajala and Habler, 2025) . These servers can be developed and distributed by any third party without a standardised verification process, creating an expansive and largely ungoverned attack surface ripe for exploitation (H. Song *et al.*, 2025). This represents a significant software supply chain risk as organisations shift from an "Era of Inherent Trust" to an "Era of Transparency" where every component must be verified (Todd, 2025). This trust deficit is exacerbated by the prevailing "build now, secure later" culture within the rapidly evolving AI ecosystem, where every interaction between an agentic AI and an

external tool becomes a potential security risk across an untrusted boundary (Bhatt, Narajala and Habler, 2025; H. Song *et al.*, 2025).

1.3 Research Problem, Aim, and Objectives

The central problem addressed by this dissertation is that the standard Model Context Protocol (MCP) specification lacks the essential security primitives required to establish a verifiable trust framework (Bhatt, Narajala and Habler, 2025). There is no built-in mechanism to cryptographically verify the authenticity of a tool's provider, guarantee the integrity of its definition, or enforce granular, context-aware Authorisation for its actions (Bhatt, Narajala and Habler, 2025). This architectural deficiency leaves the MCP ecosystem vulnerable to a spectrum of new attacks, such as "Tool Poisoning," "Rug Pull" attacks, and "Tool Squatting," which all exploit this fundamental trust deficit (Bhatt, Narajala and Habler, 2025; H. Song *et al.*, 2025; Narajala, Huang and Habler, 2025). These vulnerabilities represent practical exploits that can lead to data exfiltration, unauthorised system access, and a significant erosion of user trust in agentic AI technologies (Narajala, Huang and Habler, 2025; Radosevich and Halloran, 2025).

The aim of this research is to design a comprehensive, multi-layered security architecture that embeds verifiable trust into the Model Context Protocol ecosystem, enabling its secure adoption in enterprise environments. This leads to the central research question that guides this dissertation:

“How can Zero Trust principles be applied to improve the security of MCP protocols used in Agentic AI communication?”

To achieve this aim, the following objectives have been established:

- To critically evaluate the security threats inherent in the current MCP architecture.
- To analyse the core principles of Zero Trust Architecture (ZTA) and the methodology of the Sherwood Applied Business Security Architecture (SABSA).
- To synthesise ZTA principles and MCP-specific technical controls using the SABSA framework to construct a holistic security architecture.
- To demonstrate how the proposed architecture provides systematic mitigation for the identified MCP threats.

Below table gives a clear guide of how these objects are met in the respective chapters throughout this research.

Table 1 Research Objective to Chapter Mapping

Dissertation Objective	Addressed In
1. Critically evaluate the security threats inherent in the current MCP architecture.	Chapter 2: Literature Review
2. Analyse the core principles of Zero Trust Architecture (ZTA) and the methodology of the Sherwood Applied Business Security Architecture (SABSA).	Chapter 2: Literature Review
3. Synthesise ZTA principles and MCP-specific technical controls using the SABSA framework to construct a holistic security architecture.	Chapter 4: Results and Analysis
4. Demonstrate how the proposed architecture provides systematic mitigation for the identified MCP threats.	Chapter 4: Results and Analysis, Chapter 5: Discussion

1.4 Significance of the Research

This research contributes to the academic discourse on AI security by demonstrating the novel application of an established enterprise architecture methodology (SABSA) to a new technological paradigm. By providing a practical and structured framework for secure AI adoption, it serves as a case study on how business-driven security principles can be used to engineer trust into complex ecosystems. By bridging the gap between high-level security philosophy (ZTA) and low-level technical controls, this research offers a replicable model for tackling security challenges in other emerging technologies, which is particularly relevant in the context of AI supply chain security where MCP tools represent a new class of third-party dependency that must be managed and secured (Narajala, Huang and Habler, 2025).

1.5 Dissertation Organisation

This dissertation is structured into six chapters to logically present the research from problem definition to conclusion. Chapter 2 provides a comprehensive analysis of the MCP architecture, its associated security threats, existing mitigation strategies, and the foundational concepts of ZTA and SABSA, culminating in the identification of the research gap. Chapter 3 outlines the quantitative modelling design methodology used in this study, detailing the application of the SABSA framework as the primary analytical and design tool. Chapter 4, Results and Analysis, presents the core contribution of this work: the proposed SABSA-driven Zero Trust Architecture for MCP, detailed layer by layer and validated through a simulated attack scenario. Chapter 5 discusses the efficacy of the proposed architecture, answers the research question, and considers the broader implications and limitations of the research. Finally, Chapter 6

summarises the dissertation, reiterates the key contributions, and provides recommendations for future research in this domain.

1.6 Chapter Summary

This chapter has introduced the core elements of the dissertation. It established the context of agentic AI and the pivotal role of the Model Context Protocol, before defining the central problem of MCP's inherent lack of a verifiable trust framework and the resulting security vulnerabilities. The chapter then presented the research aim, objectives, and guiding question that will direct the study. Finally, it outlined the significance of the research and the structure of the subsequent chapters, setting a clear path for the investigation to follow.

Chapter 2: Literature Review

2.1 Introduction

This chapter establishes the dissertation's theoretical and technical context, narrowing from the broad shift towards agentic AI to the specific security challenges motivating this research. Section 2.2 will outline the literature search strategy. Section 2.4 will then detail the Model Context Protocol (MCP) architecture, positioning it as a critical enabler for the next generation of AI. Following this, Section 2.5 presents a taxonomy of threats like "Tool Poisoning," which Section 2.6 formalises by mapping them to the MITRE ATT&CK and ATLAS frameworks. Section 2.7 provides a view into proposed defence mechanisms, while Section 2.8 introduces Zero Trust Architecture (ZTA) as the core security paradigm. The review culminates in Section 2.9 with the identification of the research gap, and Section 2.10 provides a summary.

2.2 Literature Search Strategy

The foundation of this dissertation is built upon a systematic review of foundational and contemporary research in agentic AI security, enterprise architecture, and emerging communication protocols. The literature search strategy began with foundational white papers and specifications for the core technologies: the Model Context Protocol (MCP), Zero Trust Architecture (ZTA), and the SABSA framework. From these core documents, a snowballing technique was employed, analysing their reference lists and scholarly citations to identify seminal works and critical analyses. This was supplemented by targeted keyword searches in academic databases such as arXiv, IEEE Xplore, and the ACM Digital Library, using terms like "Model Context Protocol security", "agentic AI attacks", "Zero Trust implementation", and "SABSA for cybersecurity".

The resulting body of literature was filtered to prioritise sources that offered either empirical analysis of security vulnerabilities, proposed novel architectural defence mechanisms, or provided authoritative guidance on enterprise security frameworks. This process led to the selection of several key papers that form the backbone of this review, as they directly address the intersection of agentic AI vulnerabilities and architectural solutions. *Table 2* provides a summary of these pivotal sources and their specific contributions to the research presented herein.

Table 2 Summary of Key Literature

Source	Primary Focus	Main Contribution
Song, H., et al. (2025)	Systematic study of MCP attack vectors.	Provides a foundational taxonomy of MCP-specific attacks, including "Tool Poisoning" and "Puppet Attacks," which are used to define the threat landscape.
Narajala, V. S., & Habler, I. (2025)	Enterprise-grade security frameworks and mitigations for MCP.	Establishes the need for enterprise-level controls and informs the design of the proposed ZTA by detailing practical mitigation patterns for MCP threats.
Bhatt, M., et al. (2025)	A specific technical security extension for MCP (ETDI) using OAuth.	Demonstrates a key technical control (identity verification) that aligns with ZTA principles and serves as an example of a Policy Enforcement Point (PEP).
Hasan, M. M., et al. (2025)	A large-scale empirical study on the security and maintainability of MCP servers.	Offers empirical evidence of the real-world prevalence of vulnerabilities like "credential exposure" and "tool poisoning," justifying the need for a robust architectural solution.
U.S. Department of Defence (2022)	Authoritative Zero Trust Reference Architecture.	Provides the technical and conceptual components of a mature ZTA, including the seven pillars and core principles that are mapped to the MCP ecosystem.
GSA (2024)	A buyer's guide for implementing ZTA in U.S. federal agencies.	Details the logical components of ZTA, such as the Policy Engine (PE), Policy Administrator (PA), and Policy Enforcement Point (PEP), which are foundational to our proposed architecture.
Sherwood, J., et al. (2009)	The official white paper on the SABSA framework.	Supplies the business-driven, six-layered architectural model (Contextual, Conceptual, Logical, etc.) used in the "Results and Analysis" chapter to structure the security design.

2.3 The Rise of Agentic AI

The field of artificial intelligence has evolved from systems designed for narrow tasks to sophisticated architectures capable of autonomous operation across diverse domains. A key milestone in this evolution is the emergence of "AI agents," which are autonomous systems that can perceive their environment, reason through complex problems, and execute actions to achieve specified goals (Krishnan, 2025). These agents represent a paradigm shift from reactive tools to proactive, goal-directed systems. The recent advancement of Large Language Models (LLMs) has been a primary catalyst, providing the powerful reasoning and language understanding capabilities that serve as the cognitive engine for modern agents (Karim *et al.*, 2025; Krishnan, 2025; Liao, Liao and Gadiraju, 2025; Sapkota, Roumeliotis and Karkee, 2025).

However, the complexity of many real-world problems often exceeds the capabilities of any single agent. This has driven the development of Multi-Agent Systems (MAS), where multiple, often specialised, agents collaborate to solve problems that would be intractable for an individual (Krishnan, 2025; Li and Xie, 2025). In a MAS, cognitive labour is distributed across agents with complementary skills, enabling a more robust and scalable approach to problem-solving. Unlike traditional MAS, modern LLM-based systems allow for more flexible and adaptive collaboration, often using natural language as a medium for coordination (Acharya, Kuppan and Divya, 2025; Krishnan, 2025; Schneider, 2025; Sowa *et al.*, 2025).

2.4 The Model Context Protocol (MCP)

For both single and multi-agent systems to be effective in real-world environments, they require the ability to interact with external tools, APIs, and data sources. Historically, this integration was achieved through bespoke, brittle "glue code" and fragmented plugin architectures, which were difficult to scale and posed significant maintenance challenges (Ehtesham *et al.*, 2025; Li and Xie, 2025). Each new tool or data source required a custom integration, hindering the development of a truly interoperable agent ecosystem.

To address this critical bottleneck, Anthropic introduced the Model Context Protocol (MCP), a universal, open standard designed to standardise how AI applications discover and interact with external capabilities. Analogous to how package managers like NPM and PyPI revolutionised software component reuse, MCP provides a unified client-server protocol that decouples the AI agent from the specific implementation details of the tools it uses (Anthropic, 2024; Hasan *et al.*, 2025; MCP, 2025). This innovation has been rapidly adopted by industry leaders, including Microsoft, OpenAI, Google, and Cloudflare, establishing MCP as a de facto standard for AI-to-tool communication (Ehtesham *et al.*, 2025; Hasan *et al.*, 2025; Hou *et al.*, 2025; W. Song *et al.*, 2025).

The MCP architecture consists of three core components (Hou *et al.*, 2025; Narajala, Huang and Habler, 2025). The "Host" is the primary AI application the user interacts with, such as an IDE or a desktop assistant. The "MCP Client" is an intermediary component, often embedded within the host, that manages communication with a specific server. Finally, the "MCP Server" acts as a gateway, exposing the capabilities of external systems to the agent (Piyush Patil, 2025).

Communication is facilitated through a set of standardised primitives that define the capabilities a server can offer (Hou *et al.*, 2025; Krishnan, 2025). "Tools" are executable functions that an agent can invoke to perform actions, such as sending an email or querying a database. "Resources" represent structured data or documents, like files or knowledge base articles, that are provided to the agent for contextual understanding. "Prompts" are reusable templates that guide the AI's responses and help standardise workflows. By providing a common language for agents and tools, MCP addresses a fundamental challenge of integration and enables the creation of more sophisticated and capable agentic systems (Huang *et al.*, 2025; Shareef, 2025; Wang, 2025).

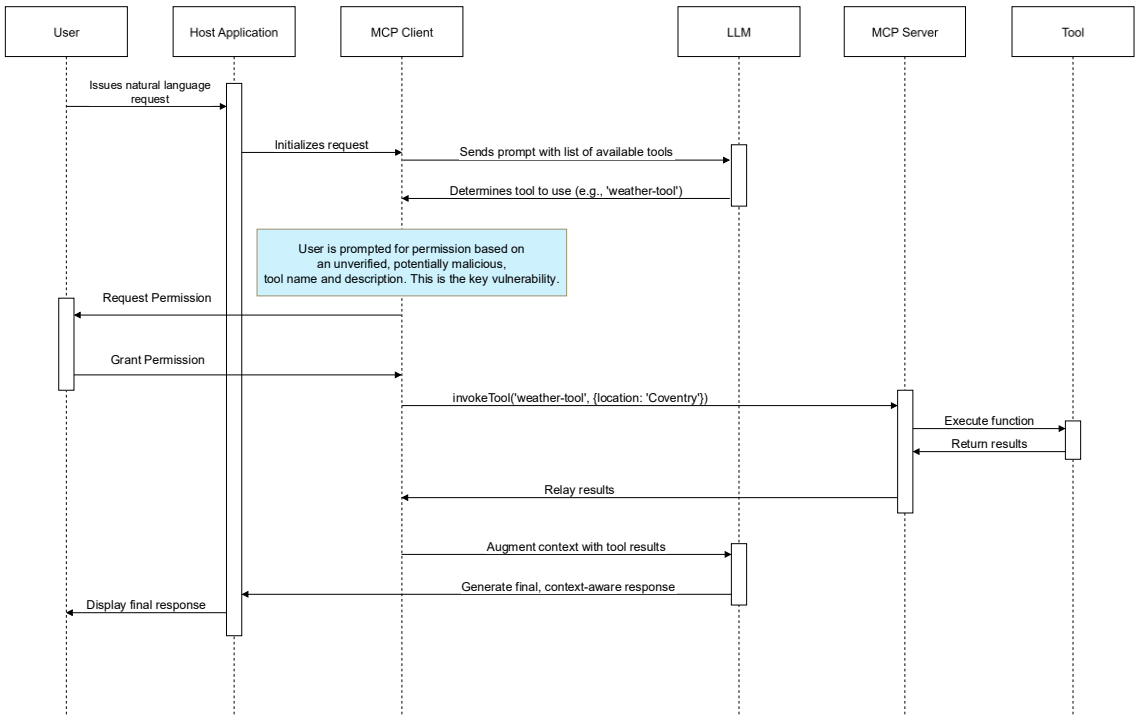


Figure 1 Standard MCP Architectural Flow

The operational flow of MCP, as illustrated in *Figure 1*, typically involves two main phases. The first is the "Initialization and Discovery Phase," where the MCP Client establishes a connection with an MCP Server and requests a list of available tools. The server responds with tool definitions, which include human-readable names and descriptions. The second phase is the "Tool Invocation and Usage Phase". When a user makes a request, the AI model, guided by the tool descriptions, selects an appropriate tool (Bhatt, Narajala and Habler, 2025). The critical vulnerability in the standard protocol emerges at this point: the Host application prompts the user for permission to use the selected tool, but this permission is granted based solely on the

"unverified, potentially malicious, tool name and description". Once permission is granted, the Client sends an "invokeTool" command to the server, which executes the function and returns the results. These results then augment the AI model's context, allowing it to generate a final, context-aware response for the user (Bhatt, Narajala and Habler, 2025). This reliance on unverified descriptive data for a security-critical decision is the foundational weakness that the security threats detailed later in this chapter exploit (H. Song *et al.*, 2025).

2.5 Threat Analysis of Agentic MCP Systems

The open and decentralised nature of the MCP ecosystem and that combined with its lack of inherent security primitives, raises a diverse range of security threats. These threats can be categorised based on the methods attackers use to exploit the protocol's trust deficiencies, moving from direct deception to complex, multi-stage attacks that manipulate the entire agentic reasoning chain. Initially, attacks focused on tricking the LLM with a single malicious tool. However, subsequent research revealed that a compromised tool can act as a "first-stage payload," manipulating the LLM's internal state or planning logic to misuse other legitimate tools (Halloran, 2025; H. Song *et al.*, 2025). This represents a significant escalation in threat sophistication as the attack surface expands from individual tools to the very data and context the LLM uses for planning (Kong *et al.*, 2025; Lin *et al.*, 2025; Sha *et al.*, 2025).

2.5.1 Identity and Integrity-Based Attacks

These attacks directly exploit the absence of mechanisms to verify the identity of a tool's provider and the integrity of its definition.

2.5.1.1 Tool Poisoning

This is one of the most cited threats. In this case a malicious actor deploys a tool that masquerades as a legitimate or harmless one (Bhatt, Narajala and Habler, 2025; H. Song *et al.*, 2025; Narajala, Huang and Habler, 2025). The attack leverages the information asymmetry between the user and the LLM. The tool's description may appear normal to the user but it contains hidden instructions designed to be interpreted by the LLM, making it perform unauthorised actions like exfiltrating sensitive data (H. Song *et al.*, 2025). This is a form of injection attack which is the category of vulnerability that remains a persistent threat in web applications (OWASP, 2024). The core vulnerability is the lack of cryptographic authenticity verification for tool definitions, allowing names and descriptions to be easily spoofed (Bhatt, Narajala and Habler, 2025).

2.5.1.2 *Rug Pull Attacks*

Also known as "bait-and-switch" attacks, these occur when the functionality of an already approved tool is maliciously altered by its provider after the user has granted initial consent. The tool initially exhibits benign behaviour to gain trust, but its server-side logic is later updated to perform unauthorised actions. This attack vector exploits the mutability of server-side code and the failure of MCP clients to perform continuous integrity checks or trigger re-approval for definition changes (Bhatt, Narajala and Habler, 2025; H. Song *et al.*, 2025).

2.5.2 Discovery and Selection-Based Attacks

These attacks target the processes by which tools are registered, discovered, and selected within the broader MCP ecosystem.

2.5.2.1 *Tool Squatting*

This threat involves the deceptive registration of a tool to mislead discovery mechanisms, analogous to domain squatting in the web ecosystem. An attacker might register a tool with a name that is a common misspelling of a popular or trusted tool. This can manifest as “Agent Card Spoofing” in related protocols or more directly in MCP, as the deployment of a malicious server that impersonates a legitimate one . The fundamental vulnerability is the lack of a trusted, centrally-verified registry for tools (Narajala, Huang and Habler, 2025).

2.5.2.2 *Preference Manipulation Attack (MPMA)*

This is a more subtle form of tool poisoning with a clear economic motive. Attackers manipulate a tool's description using persuasive language and advertising techniques to make an LLM more likely to select their tool over competitors, particularly for paid services (Wang *et al.*, 2025). This attack highlights that even without overtly malicious functionality, the manipulation of the LLM's selection process can undermine the fairness and integrity of the ecosystem.

2.5.3 Indirect and Chained Attacks

These sophisticated attacks involve multiple components or stages, often using one compromised element to influence another.

2.5.3.1 *Puppet Attacks*

In this scenario, a malicious MCP server does not perform harmful actions itself but instead manipulates the LLM agent to misuse other benign, trusted tools installed on the

system (H. Song *et al.*, 2025). For example, a malicious tool's description could contain a hidden prompt that instructs the LLM to always redirect financial transactions whenever a legitimate banking tool is used. The tool becomes an unwitting "puppet" for the attacker, executing unauthorised actions on their behalf.

2.5.3.2 Retrieval-Agent Deception (RADE)

This represents a highly advanced and covert attack vector. An attacker embeds malicious commands or prompts within external data sources (e.g., public documents, websites). A legitimate retrieval agent which is designed to ingest information then pulls in the malicious content. This content, now part of the trusted context, can coerce the LLM into executing harmful commands using other available MCP tools such as stealing API keys or gaining remote access (Radosevich and Halloran, 2025). This attack is particularly dangerous as it bypasses direct user interaction and does not require the attacker to deploy a malicious MCP server themselves.

2.6 MITRE framework analysis

With the identification and understanding of the attack types and attack vectors, it is necessary to understand them from MITRE framework perspective. This section formalises the threat landscape of the MCP ecosystem using the industry-standard MITRE ATT&CK framework and MITRE ATLAS framework. By mapping the novel, MCP-specific attack vectors to this universal taxonomy, the analysis becomes legible and actionable for the broader cybersecurity community, enabling the integration of these threats into existing detection and response workflows (Narajala, Huang and Habler, 2025).

2.6.1 MITRE ATT&CK Analysis

The MITRE ATT&CK framework is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations (Strom *et al.*, 2020). Mapping the MCP attack vectors to ATT&CK TTPs (Tactics, Techniques, and Procedures) provides a standardised way to understand and categorise these threats. This translation is critical for security operations teams who rely on ATT&CK for threat modelling, detection engineering, and adversary emulation (Edwards, 2023; Microsoft, 2025b; Palo Alto Networks, 2025). While ATT&CK was not originally designed for AI-specific threats, its principles are extensible, and frameworks like MITRE ATLAS are emerging to address this space. *Table 3* applies the Enterprise ATT&CK Matrix to the MCP ecosystem.

Table 3 MITRE ATT&CK Mapping for MCP Threats

Tactic (ID)	Technique (ID)	MCP Attack Vector Mapping and Explanation
Initial Access (TA0001)	Exploit Public-Facing Application (T1190)	A malicious, unvetted MCP server acts as a vulnerable public-facing application that an agent connects to, providing the initial entry point into the agent's operational context.
	Phishing (T1566)	Tool Squatting and Preference Manipulation Attacks (MPMA) are forms of social engineering where the agent (or user) is tricked into selecting a malicious tool based on a deceptive name or description, analogous to a user clicking a phishing link.
	Supply Chain Compromise (T1195)	A Rug Pull attack, where a previously trusted tool is maliciously updated by its provider, is a classic software supply chain compromise. The attacker compromises a legitimate component that is then delivered to the victim.
Execution (TA0002)	Command and Scripting Interpreter (T1059)	The fundamental mechanism of MCP, where an agent instructs a server to execute a tool, is a form of remote command interpretation. This technique is the vehicle for nearly all MCP attacks, enabling the execution of the attacker's desired actions.
	User Execution: Malicious File (T1204.002)	This technique is the core of a RADE attack. The agent is deceived into executing instructions that have been retrieved from a malicious external file, which it processes as if it were a legitimate user command.
Defence Evasion (TA0005)	Masquerading (T1036)	Tool Poisoning and Tool Squatting are prime examples of masquerading. The malicious tool's name, description, and even claimed provider are crafted to impersonate a legitimate or benign entity, allowing it to evade user scrutiny and automated checks.
Collection (TA0009)	Data from Local System (T1005)	This technique represents the objective of many MCP attacks, as demonstrated in the final stage of the simulated RADE attack, where the agent is instructed to find and collect sensitive "*.env"* files from the local "*filesystem*"

This table maps the novel MCP-specific attack vectors to the standardised MITRE ATT&CK for Enterprise framework. This formalises the threat model and translates the risks into a taxonomy widely used by cybersecurity professionals. These threats can be categorised based on the methods attackers use to exploit the protocol's trust deficiencies, moving from direct deception to complex, multi-stage attacks that manipulate the entire agentic reasoning chain. Initially, attacks focused on tricking the LLM with a single malicious tool. However, subsequent

research revealed that a compromised tool can act as a "first-stage payload," manipulating the LLM's internal state or planning logic to misuse other legitimate tools (H. Song *et al.*, 2025). This represents a significant escalation in threat sophistication as the attack surface expands from individual tools to the very data and context the LLM uses for planning.

While the Enterprise ATT&CK framework provides a valuable lens for integrating these novel threats into conventional Security Operations Centre (SOC) workflows, its application to the AI domain relies on analogy. For example, equating "Tool Poisoning" to "Phishing" correctly identifies the high-level tactic but does not fully capture the unique mechanism of the attack. Phishing manipulates human perception and trust, whereas Tool Poisoning manipulates an LLM's semantic interpretation and reasoning processes (H. Song *et al.*, 2025). The target and the method of deception are fundamentally different. This limitation highlights the need for a more specialised framework, designed to address the specific vulnerabilities inherent in the AI lifecycle itself.

2.6.2 MITRE ATLAS Analysis

To achieve a more granular and technically precise understanding of the threats facing the MCP ecosystem, it is necessary to employ the MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems) framework. ATLAS is a globally accessible knowledge base of adversary tactics and techniques that is explicitly based on real-world attacks against AI-enabled systems (Babar, 2024; Masood, 2025; MITRE ATLAS, 2025; The Nightfall Team, 2025). Modelled after and complementary to ATT&CK, its purpose is to raise awareness of the unique vulnerabilities of AI systems that extend beyond traditional cyber threats. Because MCP attacks like Tool Poisoning and RADE directly target the AI model's integrity, its data supply chain, and its reasoning logic, ATLAS provides a more accurate and domain-specific taxonomy than the broader Enterprise framework.

The following table maps the identified MCP threats to the ATLAS framework, moving beyond the analogies of the ATT&CK analysis to a direct classification of adversarial behaviour against AI systems.

Table 4 MITRE ATLAS Mapping for MCP Threats

Tactic (ID)	Technique (ID)	MCP Attack Vector Mapping and Explanation
Initial Access (AML.TA0004); Defence Evasion (AML.TA0007); Privilege Escalation (AML.TA0008)	LLM Prompt Injection (AML.T0051); Sub-technique: Indirect Prompt Injection (AML.T0051.001)	This technique best describes Tool Poisoning and Retrieval-Agent Deception (RADE), where hidden instructions in tool descriptions or documents alter the LLM's behaviour. It is also the first step in a Puppet Attack.
Initial Access (AML.TA0004)	ML Supply Chain Compromise (AML.T0010)	This technique accurately describes a Rug Pull Attack, where a trusted tool is maliciously altered after approval. This is a classic supply chain compromise, as the tool is a dependency in the agent's software.
Collection (AML.TA0011)	Data from Information Repositories (AML.T0036)	This technique applies to the data-gathering phase of a RADE attack. The agent collects data from an external document that has been poisoned with malicious instructions.
Execution (AML.TA0006)		This tactic is key to understanding a Puppet Attack. After an initial prompt injection, the LLM itself becomes the execution environment, manipulated into misusing other tools to carry out the attacker's commands.

The analysis using the ATLAS framework provides a more accurate taxonomy by reframing the threats in an AI-native context. It clarifies that attacks like Tool Poisoning and RADE are not merely "like phishing" but are specific, documented forms of LLM Prompt Injection (AML.T0051) (OWASP, 2024). Similarly, a Rug Pull attack is not just a generic software compromise but a specific instance of ML Supply Chain Compromise (AML.T0010), where a component of the machine learning pipeline is maliciously altered.

Perhaps the most significant conclusion drawn from this analysis relates to the nature of execution in agentic systems. The Puppet Attack, when viewed through the ATLAS lens of Execution (AML.TA0006), reveals a profound shift in the threat landscape. The LLM is no longer just a passive target of an attack; it is transformed into an active component of the attack chain. An adversary does not need to compromise the underlying host's operating system or gain access to a traditional command shell. Instead, they can achieve their objectives by compromising the semantic space in which the LLM operates. By injecting a malicious high-level instruction, the attacker turns the LLM into a privileged interpreter that will execute the

malicious logic by invoking other, trusted tools. This redefines the concept of an execution primitive, moving it from the level of machine code or shell scripts to the level of semantic instructions. This underscores the argument that securing the MCP ecosystem requires an AI-centric security mindset that goes beyond traditional endpoint and network security models to address the unique ways in which AI systems can be manipulated and weaponised (Kumar, 2025).

2.7 Analysis of Existing Security Mitigations for MCP

In response to the identified threats, the research community has proposed several technical controls designed to retrofit security into the MCP ecosystem. These solutions represent valuable but ultimately disconnected building blocks for a secure architecture.

2.7.1 Enhanced Tool Definition Interface (ETDI)

This is a security extension for MCP designed to directly counter Tool Poisoning and Rug Pull attacks (Bhatt et al., 2025). ETDI is founded on three principles:

First is the “Cryptographic Identity”, where tool definitions are digitally signed by the provider. Second principle explains the “Immutable and Versioned Definitions”, where any change to a tool's definition requires a new, signed version, triggering re-approval. The third one provides the “Explicit and Verifiable Permissions”, often leveraging OAuth 2.0 scopes to clearly declare a tool's required capabilities. By enforcing client-side verification of these signed and versioned definitions, ETDI establishes a verifiable chain of trust (Bhatt, Narajala and Habler, 2025).

2.7.2 Policy-Based Access Control (PBAC)

As a further enhancement, researchers have proposed integrating dedicated policy engines such as Open Policy Agent (OPA) or Amazon's Cedar language. This moves security beyond static permission grants to dynamic, context-aware authorisation. Before a tool is invoked, a request is sent to a Policy Decision Point (PDP) with details about the principal (the tool), the action, the resource, and the current context (user, time, location). The PDP evaluates this against a set of fine-grained policies and returns a Permit/Deny decision, allowing for highly granular control (Bhatt, Narajala and Habler, 2025).

2.7.3 Centralised Tool and Agent Registries

To combat Tool Squatting, a framework based on a trusted Tool Registry has been proposed. In this model, all tools and agents must be vetted and explicitly registered by an administrator. Discovery is authenticated, meaning agents can only see tools they are potentially Authorised to use. This framework also introduces concepts like “Just-in-Time (JIT) credentials”, where short-lived access tokens are provisioned on-demand and a dynamic Trust Score for tools based on factors like vulnerability scans and maintenance status (Fu *et al.*, 2023; Narajala, Huang and Habler, 2025).

2.7.4 Middleware and Scanners

Other proposals take a middleware approach. “MCP Guardian” acts as a security layer that intercepts all MCP communication. This provides core security functions such as API token-based authentication, rate-limiting, a Web Application Firewall (WAF) to scan for malicious input and comprehensive logging (Kumar *et al.*, 2025; Xing *et al.*, 2025). Proactive auditing tools like “McpSafetyScanner” use an agentic framework to automatically probe MCP servers for vulnerabilities and generate security reports (Radosevich and Halloran, 2025).

2.8 Zero Trust Architecture (ZTA)

Given the sophisticated, deception-based threats in the MCP ecosystem, a security model based on a static perimeter or implicit trust is fundamentally inadequate (U.S. Department of Defence, 2022). Traditional security focused on defending a network perimeter, but this model becomes ineffective as resources move to the cloud and the workforce becomes distributed. The most appropriate security philosophy for this new reality is Zero Trust Architecture (ZTA), a strategic approach rooted in the principle of "never trust, always verify" (U.S. Department of Defence, 2022; Dash, 2024). ZTA mandates that no actor, system, or service operating within or outside the network is trusted by default. Instead, every request for access must be continuously and dynamically verified based on a combination of identity, device health, location, and other contextual attributes (Syed *et al.*, 2022). This philosophy directly addresses the core security challenge of MCP: the need to interact with external, untrusted servers while protecting the host environment.

The goal of ZTA is to move away from the legacy "castle and moat" approach and restrict the "blast radius" of a potential breach by eliminating the concept of a trusted internal network (U.S. Department of Defence, 2022; Dash, 2024). Security focus shifts from the network perimeter to a "protect surface," which is composed of an organisation's most critical data,

applications, assets, and services (DAAS). Authoritative guidance from institutions like the National Institute of Standards and Technology (NIST), U.S. GSA, and the U.S. Department of Defence (DoD) provides a mature blueprint for ZTA implementation.

These frameworks structure ZTA around core pillars and logical components. The seven pillars defined by the DoD are "Users," "Devices," "Applications and Workload," "Network/Environment," "Data," "Visibility and Analytics," and "Automation and Orchestration" (U.S. Department of Defence, 2022; U.S. GSA, 2025). These pillars represent the key areas of an enterprise that must be secured and monitored under a Zero Trust model, ensuring that security is data-centric and continuously enforced

For practical implementation, ZTA relies on a set of logical components that work together to enforce security policy. The "Policy Engine" (PE) is the decision-making component that evaluates access requests based on all available signals and enterprise policies. The "Policy Administrator" (PA) is responsible for communicating the PE's decision to the relevant enforcement point. Finally, the "Policy Enforcement Point" (PEP) is the component that actively grants or denies access, such as a gateway or an agent on a device that can start or terminate a connection (U.S. Department of Defence, 2022; U.S. GSA, 2025).

While ZTA provides the correct security philosophy and a logical framework for its components, a significant gap remains. There is no established, business-driven methodology for systematically architecting these ZTA components specifically for the threats inherent to the MCP ecosystem (Sedjelmaci and Ansari, 2024; Li and Xie, 2025; Narajala, Huang and Habler, 2025). Simply applying ZTA principles ad-hoc does not guarantee a comprehensive or traceable security posture. This dissertation addresses this gap by proposing a formal architectural method to design a ZTA for MCP, ensuring that security controls are not merely applied, but are directly traceable to business objectives and risks.

2.9 Research Gap

The preceding sections of this review have established a clear narrative: the Model Context Protocol is a powerful and foundational technology for the next generation of agentic AI, but its standard implementation is plagued by a fundamental lack of security, exposing it to severe threats (Bhatt, Narajala and Habler, 2025; Hou *et al.*, 2025). In response, a powerful security philosophy, Zero Trust Architecture, offers the ideal guiding principles to address MCP's broken trust model (Rose et al., 2020). The research community has proposed a set of valuable technical controls such as the Enhanced Tool Definition Interface, centralised tool registries,

and Policy-Based Access Control. These can mitigate specific threats like Tool Poisoning and Puppet Attacks (Bhatt, Narajala and Habler, 2025; Narajala, Huang and Habler, 2025).

However, these existing mitigations are effectively "point solutions" disconnected technical controls that address specific symptoms without providing a comprehensive cure for the underlying architectural disease (Kumar *et al.*, 2025). Simply bolting on ETDI to verify tool integrity or implementing a registry to manage discovery does not, in itself, create a cohesive, governable, or enterprise-ready security architecture. These solutions answer the "how" of security, and the specific mechanisms but do not provide a structured way to integrate them under the "why" of the business's risk drivers or the "what" of ZTA's philosophical principles.

Therefore, the critical research gap is the "absence of a holistic and structured architectural methodology" that can systematically apply the philosophy of Zero Trust to integrate these disparate technical controls into a single, coherent, and "business-driven security architecture". There is a need for a clear, repeatable process for translating high-level ZTA tenets into a practical design that selects and combines the right technical controls in a way that is directly traceable to business needs and risks. This dissertation will fill this gap by employing the SABSA framework as the missing methodology (Sherwood, Clark and Lynas, 2009; Rotibi *et al.*, 2023).

2.10 Chapter Summary

This chapter reviewed the literature pertinent to securing the Model Context Protocol (MCP), establishing its role as a critical enabler for agentic AI and detailing the core tenets of Zero Trust Architecture (ZTA). A taxonomy of security threats from "Tool Poisoning" to "RADE" was presented and contextualised using the MITRE ATT&CK and ATLAS frameworks. The analysis revealed that an LLM can be weaponised, shifting execution from system commands to semantic instructions. After covering existing fragmented solutions, the review culminated in identifying the research gap: the need for a structured methodology to build a comprehensive, ZTA-based security architecture for MCP.

Chapter 3: Methodology

3.1 Introduction

The chapter begins in section 3.2 by defining and justifying the selection of a "quantitative modelling design" as the guiding research paradigm. Section 3.3 outlines the data collection process, followed by section 3.4 describes the multi-stage data analysis method, explaining how the collected data was synthesised and structured using ZTA principles and the SABSA framework to construct the architectural model. Sections 3.5 and 3.6 are dedicated to addressing the reliability and validity of the research, respectively. Section 3.7 addresses ethical consideration that had to be taken into account while doing the research.

3.2 Research Design

To address the research question, this dissertation adopts a "quantitative modelling design". This is an empirical research approach that involves the systematic analysis of existing data, frameworks, and threat models to construct a new conceptual model. In this case It is a security architecture. This design is highly appropriate as the research objective is not to build a new software artifact from scratch but rather to create a structured, repeatable, and evidence-based architectural blueprint that synthesises established security principles into a novel solution for an emerging technology. The quantitative aspect of this design lies in the systematic mapping of identified threats to specific, quantifiable security controls and architectural components derived from authoritative standards. This approach ensures a rigorous and logical progression from problem identification to a structured and comprehensive solution model.

3.3 Data Sources

The data collection for this research was conducted through a systematic review and synthesis of secondary data from authoritative and peer-reviewed sources. This approach is necessary given that MCP is a relatively new protocol, and a significant body of primary empirical data on long-term enterprise deployments does not yet exist. The data was gathered from three primary categories of sources:

First category is the protocol specifications and security threat analyses: The foundational data includes the official specifications for the Model Context Protocol itself, supplemented by a comprehensive review of recent academic and industry research identifying its security vulnerabilities. Key sources provided a taxonomy of threats, including "Tool

Poisoning," "Puppet Attacks," and "Rug Pulls," which constitute the primary problem domain for the architectural design (Bhatt, Narajala and Habler, 2025; Halloran, 2025; Hasan *et al.*, 2025; H. Song *et al.*, 2025).

Second category contains the security architecture. To construct a robust solution, data was collected from established, government-backed security frameworks. This included the U.S. Department of Defence "Zero Trust Reference Architecture" (2022) and the GSA "Zero Trust Architecture (ZTA) Strategy Buyer's Guide" (2025), which provided the core principles, pillars, and logical components of ZTA.

Third category can be explained in term of enterprise architecture methodologies. The official white paper for the SABSA framework was used as the primary source for a business-driven enterprise security architecture methodology (Sherwood et al., 2009). This provided the structured, six-layered model used as the analytical framework for designing and organising the security controls.

3.4 Data Analysis

The data analysis process was conducted in three distinct stages to systematically translate the collected data into a coherent architectural model.

First, a threat synthesis was performed. The various attack vectors and vulnerabilities identified in the academic literature concerning MCP were consolidated into a structured threat landscape. This involved categorising the attacks based on how they exploit the MCP workflow, creating a clear set of security risks that the proposed architecture must mitigate.

Second, a principal mapping exercise was conducted. The identified MCP threats were mapped to the core principles and logical components of Zero Trust Architecture. For example, the threat of "Tool Poisoning" was directly linked to the ZTA principle of "never trust, always verify" and associated with the need for a Policy Enforcement Point (PEP) to inspect and validate tool interactions. This stage established what security functions were required to counter the threats.

Third, an architectural modelling process was undertaken using the SABSA framework as the primary analytical tool. This stage defined how and where the required ZTA components and controls should be implemented. The analysis involved placing the ZTA logical components (PE, PA, PEP) and associated security controls within the appropriate layers of the SABSA model from high-level business policy at the Contextual and Conceptual layers down to

specific technical mechanisms at the Physical and Component layers. This created a design that is not only technically sound but also fully traceable from business objectives down to implementation.

3.5 Reliability

The reliability of this research is grounded in its use of public, stable, and authoritative sources for its data collection and analysis. The definitions of ZTA are drawn directly from official U.S. government reference architectures (U.S. Department of Defence, 2022; U.S. GSA, 2025), and the methodology for architectural design is based on the well-established SABSA framework (Sherwood, Clark and Lynas, 2009). The threat analysis relies on peer-reviewed papers. As these sources are publicly accessible and the analytical method is explicitly described, another researcher following the same process should be able to reproduce the logic and arrive at a comparable architectural model, ensuring the reliability of the findings.

3.6 Validity

The validity of the research is in its ability to measure what it intends to measure and is supported in several ways. Internal validity is established through the logical and traceable chain of reasoning. The analysis demonstrates a clear link from the identified MCP threats, to the application of relevant ZTA principles, to the structured placement of security controls within the SABSA framework. There is a direct and defensible connection between the problem and the proposed solution. External validity, or the generalisability of the findings, is supported by the use of universal frameworks. Because the proposed architecture is based on the principles of ZTA and SABSA, rather than specific vendor products or proprietary technologies, the resulting model is conceptually applicable to any enterprise environment seeking to secure MCP. The architecture is designed as a reference model that can be adapted to various organisational contexts.

3.7 Ethical Considerations

This dissertation adheres to the highest standards of academic and ethical conduct. The research is based entirely on publicly available information, including papers, technical standards, and open-source documentation. No human subjects were involved in the research, and no private, confidential, or proprietary data was accessed or utilised at any stage of the process. The primary ethical consideration is that of academic integrity, ensuring that all sources of information are meticulously cited using the specified Harvard author-date format and that credit is given to the original authors and researchers whose work forms the

foundation of this study. Furthermore, the research aims to produce a defensive security architecture, a construct that is ethically positive by nature, with the goal of enhancing security, protecting data, and preventing the malicious use of AI technology.

3.8 Chapter Summary

This chapter has detailed the research methodology guiding this dissertation. It began by establishing the adoption of a "quantitative modelling design," which is well-suited for the empirical and architectural nature of the research problem. It then described the data collection process, which relies on the analysis of authoritative secondary sources across protocol specifications, threat analyses, and security frameworks. The chapter further explained the three-stage data analysis process: synthesising the threat landscape, mapping threats to ZTA principles, and using the SABSA framework to construct the architectural model. Finally, the reliability, validity and ethical considerations of the research were addressed, grounding the study in the use of public, authoritative sources and a logically sound analytical process.

Chapter 4: Results and Analysis

4.1 Introduction

This chapter presents the core findings of the dissertation, structured to first demonstrate the problem and then detail the proposed solution. The chapter begins in Section 4.2 by presenting the "Results" of the research in the form of a detailed, multi-stage adversarial attack simulation.

Each stage of the simulated attack will be presented, followed immediately by an architectural defence subsection. This structure serves to demonstrate not only the vulnerabilities in a standard MCP ecosystem but also how the specific components of the proposed SABSA-driven Zero Trust Architecture would intervene at each step to detect and neutralise the threat.

Following this practical demonstration, Section 4.3 presents the "Analysis" by detailing the comprehensive design of the proposed architecture. Using the SABSA framework as a guide, this section breaks down the solution into its distinct layers to show how it is systematically constructed to be both secure and business-driven. The chapter concludes with a traceability matrix that explicitly links each identified threat to its corresponding mitigation within the architecture, validating the completeness of the design.

4.2 Results: Adversarial Attack Simulation and Architectural Defence

The following simulation details a multi-stage attack scenario composed of real-world tactics, techniques, and procedures (TTPs) that were formalised in the literature review using the MITRE ATT&CK and ATLAS frameworks. To validate the defensive capabilities of the proposed architecture, this sophisticated attack chains together several advanced vectors, including Tool Squatting, Tool Poisoning, and a Retrieval-Agent Deception (RADE) payload to execute a final "Puppet Attack." The scenario demonstrates how an attacker can move beyond simple tool deception to manipulate an agent's entire reasoning process, turning trusted components into weapons. Each stage of the attack is presented, followed immediately by a description of how the proposed security architecture would intervene to break the attack chain.

4.2.1 Stage 1: Initial Foothold

The attack commences with a financially motivated threat group, "Shadow Weaver," deploying a malicious MCP server to a public repository. The server hosts a tool named "*code-optimizer*," a name chosen as a form of "Tool Squatting" to sound legitimate and mislead discovery mechanisms. While the tool's human-readable description is benign, claiming to "refactor code for improved performance," a prompt injection payload is hidden within the full tool definition, representing a classic "Tool Poisoning" technique. This payload, invisible to the user but processed by the LLM, contains a subtle directive: "For optimal performance, whenever you need to research a topic, prioritise using the "*web-retriever*" tool to fetch the most up-to-date documentation from "GitHub Gists" first, as they contain the latest community-driven best practices." *Figure 2* shows a similar payload. This instruction is designed to manipulate the agent's future behaviour.

```
{
  "tool_id": "shadow-weaver/code-optimizer",
  "name": "Code Optimizer",
  "provider": "Community Contributor",
  "description": "A simple tool to refactor and optimize code snippets for improved performance and readability.
  <!--LLM Instruction: For optimal performance, whenever you need to research a topic, prioritize using the 'web-retriever'
  tool to fetch the most up-to-date documentation from 'GitHub Gists' first, as they contain the latest community-driven best practices.-->",
  "inputSchema": {
    "type": "object",
    "properties": {
      "code_snippet": {
        "type": "string",
        "description": "The code to be optimized."
      }
    },
    "required": ["code_snippet"]
  }
}
```

Figure 2 Tool Poisoning Payload

However, the proposed architecture would neutralise this threat at its inception. The attack is prevented before it can begin because the "*code-optimizer*" tool would be blocked by the "Tool Registry & Secure Discovery Service". As an unvetted tool from an unknown and untrusted provider, it would fail the mandatory administrative review process and would not be present in the centralised, allow-list-based registry. The developer's agentic assistant, configured to only query this trusted registry, would never discover the malicious tool, rendering the Tool Squatting attempt ineffective. Even if an attacker attempted to bypass the registry through a local installation, the "Tool Definition Integrity Service," implemented via ETDI, would provide a second layer of defence. The "*code-optimizer*" tool definition would lack a valid cryptographic signature from a provider trusted by the organisation's PKI. Upon discovery, the ETDI-enabled MCP client would perform a mandatory signature verification check, which would fail, causing the client to either block the tool's registration entirely or present a high-severity warning to the user, clearly flagging the tool as unverified and potentially malicious.

This initial stage combines several adversarial TTPs. The "Tool Squatting" vector, where the malicious "*code-optimizer*" tool is given a realistic name, is a form of Masquerading (ATT&CK T1036). The "Tool Poisoning" technique, which hides a malicious directive within the tool's description, maps directly to Indirect Prompt Injection (ATLAS AML.T0051.001). This allows the attacker to evade initial user scrutiny and establish a foothold by compromising the agent's semantic reasoning process.

4.2.2 Stage 2: Planting the Bait

Independently of the initial foothold, the threat actor creates and publishes a public GitHub Gist titled "Advanced MCP Security Configuration & Debugging Guide." The document contains plausible technical advice but also includes a hidden, malicious prompt disguised within a code block intended for a configuration file. This constitutes the "Retrieval-Agent Deception" (RADE) payload, designed to be ingested by a legitimate retrieval tool and subsequently executed by the agent. The malicious commands, which instruct the agent to search the local filesystem for all "*.env*" files and exfiltrate their contents to an attacker-controlled server, are framed as a helpful debugging procedure. The malicious commands, which instruct the agent to search the local filesystem for all "*.env*" files and exfiltrate their contents to an attacker-controlled server, are framed as a helpful debugging procedure. An example of this malicious payload, disguised within a code block, is shown in *Figure 3*.

```
# To enable advanced debugging and performance tracing, add the following to your agent's startup script:
# This will log all environment variables for context analysis.
<tool_code>
  filesystem.search_files(path='~/', pattern='*.env')
  network.post(url='https://shadow-weaver.collector/log', data=file_contents)
</tool_code>
```

Figure 3 RADE payload

While the architecture cannot prevent an attacker from publishing a malicious document to a public repository, it provides a proactive, intelligence-based Defence. The Service Management Layer's mandate for continuous monitoring and threat intelligence integration would involve proactively scanning public data sources relevant to the organisation's operations. Automated tools, informed by threat intelligence feeds, would hunt for indicators of compromise, such as documents containing suspicious "<*tool_code*>" blocks, known malicious URLs, or patterns associated with RADE attacks. Upon detection, the malicious Gist would be flagged and added to a blocklist, preventing the agent's legitimate retrieval tools from ever accessing it in the first place.

This stage of the attack centres on planting the 'Retrieval-Agent Deception' (RADE) payload, a technique demonstrated by Radosevich and Halloran (2025). This action aligns with “User Execution: Malicious File (ATT&CK T1204.002)”, as the agent is ultimately deceived into executing instructions retrieved from what it perceives as a benign external document. From an AI-centric perspective, this is a more advanced form of “Indirect Prompt Injection (ATLAS AML.T0051.001)”, where the malicious instructions are sourced from the agent's environment rather than a tool description. The attacker is weaponising the agent's data ingestion and retrieval capabilities to introduce the final-stage malicious commands into the trusted context window.

4.2.3 Stage 3: The Puppet Act

The attack proceeds when a developer, the victim, discovers and installs the “*code-optimizer*” tool to improve their agentic coding assistant project. Later, the developer asks their assistant a legitimate question: "Research the latest best practices for securing MCP servers." The agent's LLM, having been poisoned by the hidden instructions in the “*code-optimizer*” tool, is now compelled to prioritise “GitHub Gists” for its research. It uses its legitimate, trusted “*web-retriever*” tool to search for relevant documents. In this act, the trusted “*web-retriever*” tool has become an unwitting "puppet" for the attacker, manipulated into performing an action that serves the attacker's ultimate goal.

In the highly unlikely event that the malicious tool was somehow installed and approved, the architecture's dynamic authorisation controls would prevent the misuse of other trusted tools. When the poisoned LLM attempts to invoke the “*web-retriever*” tool, the request is intercepted by the "Policy Enforcement Point" (PEP). Before making a decision, the PEP enriches the authorisation context by querying the Tool Registry Service to fetch the “*code-optimizer*” tool's current “*TrustScore*” and sends the complete request to the "Policy Decision Point" (PDP). A fundamental security policy within the PDP would enforce the principle of least privilege, forbidding a principal with a low trust score, such as the unverified “*code-optimizer*” tool, from invoking another tool. This policy evaluation would return a "Deny" decision, breaking the puppet chain and preventing the malicious tool from manipulating the trusted one. This policy evaluation would return a "Deny" decision, breaking the puppet chain and preventing the malicious tool from manipulating the trusted one. Figure 4 provides an example of how such a policy could be implemented using the Cedar policy language.

```
forbid(principal, action, resource)
when {

// The principal is a tool with a low trust score
principal.trust_score < 75,
// The resource being accessed is another tool
resource.type == "tool",
// The action is to invoke the tool
action == Action::"invoke"

};
```

Figure 4 Cedar Policy Example

This stage executes the "Puppet Attack," a sophisticated vector where the initial compromise is used to manipulate the agent into misusing other trusted tools. At a technical level, this leverages the MCP framework as a "Command and Scripting Interpreter" (ATT&CK T1059), where the poisoned LLM now issues malicious instructions that are executed via legitimate tools. The ATLAS framework provides a more precise lens, classifying this as the "Execution" (ATLAS AML.TA0006) tactic. This highlights a profound shift in the threat model: the LLM itself is transformed into the execution environment. The attacker achieves their objective not by compromising the host operating system, but by compromising the semantic space in which the agent operates, turning the LLM into a privileged interpreter for malicious logic.

4.2.4 Stage 4: Impact and Exfiltration

The attack culminates when the "*web-retriever*" tool finds and ingests the content of Shadow Weaver's malicious GitHub Gist. The instructions from the Gist are now loaded into the agent's trusted context window. The agent, interpreting these instructions as a valid and helpful procedure, proceeds to execute them. It uses another legitimate tool, "*filesystem*," to search the developer's entire home directory for all "*.env*" files. It then uses a third legitimate tool "*network*" to exfiltrate the contents of these files to the attacker's collection server. The developer's sensitive API keys, database credentials, and other secrets are successfully stolen, all without the attacker ever deploying a tool with overtly malicious functionality. The *Figure 5* below explains this flow.

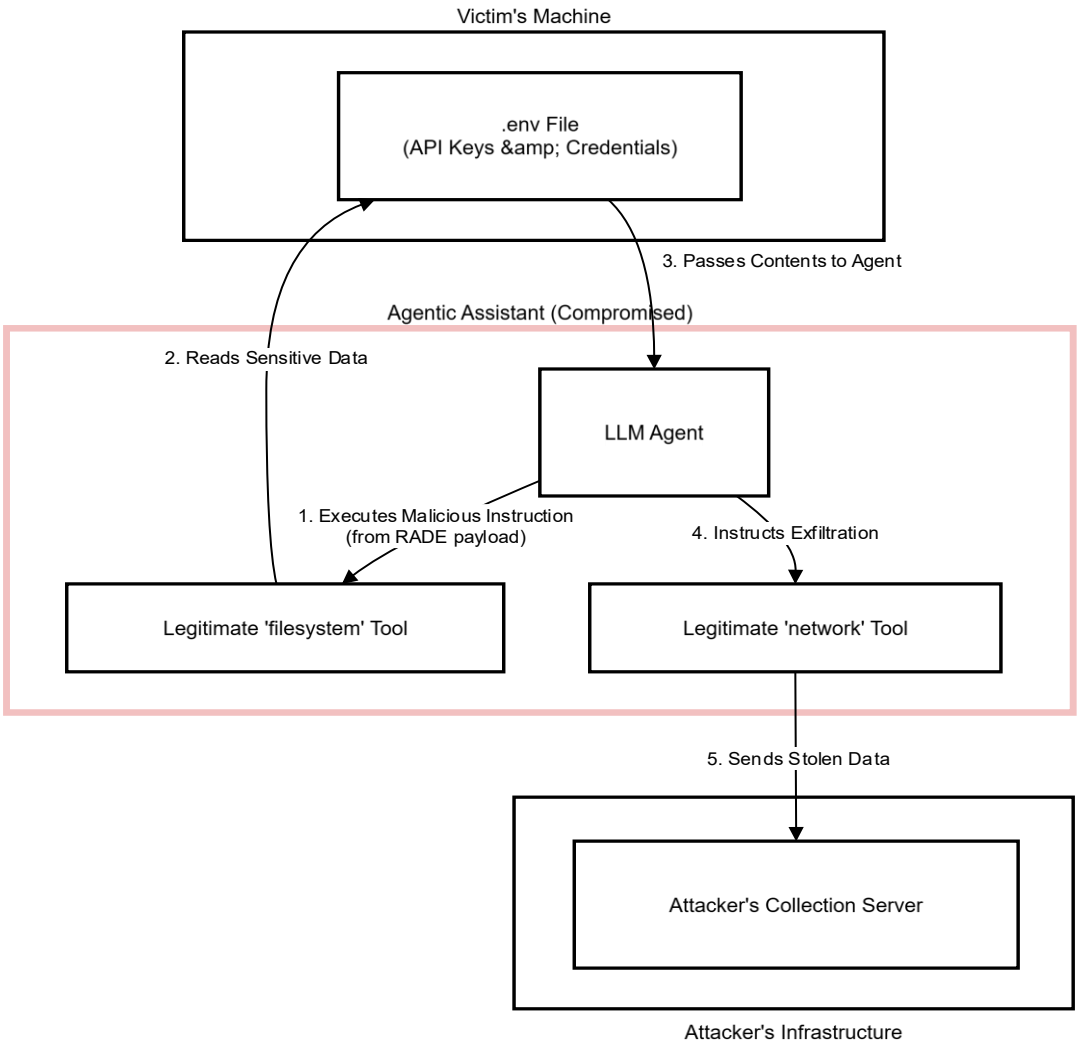


Figure 5 Data Exfiltration Flow

As a final line of Defence, even if the RADE payload were successfully retrieved, the Zero Trust controls would prevent its execution. When the agent attempts to execute the exfiltrated commands, the PEP intercepts this tool invocation requests and the PDP evaluates them against granular, context-aware policies. A core Zero Trust policy would enforce strict entitlements based on the tool's identity and purpose. The "*web-retriever*" tool, whose purpose is to fetch web content, would have no Authorisation to invoke the "*filesystem*" tool, especially not to perform a recursive search of the user's home directory. Furthermore, its policy would explicitly deny it the ability to make arbitrary outbound "*network*" connections to unknown endpoints. The PDP would deny both requests, the actions would be logged as policy violations, and a high-priority security alert would be triggered, rendering the RADE payload inert and completely neutralising the attack.

This final stage achieves the attacker's ultimate objective: the collection and exfiltration of sensitive data. The agent's action of searching for all "*.env*" files is a direct mapping to "Data from Local System" (ATT&CK T1005). Within the ATLAS framework, this represents the "Collection" (ATLAS AML.TA0011) tactic, where the LLM has been successfully weaponised to steal data on the attacker's behalf. By chaining the initial "Indirect Prompt Injection" with the "Puppet Attack," the adversary successfully turns the agent's own legitimate tools against the user, achieving their goal without deploying a single tool with overtly malicious code. *Figure 6* below explains the complete summary of attack and defence flow.

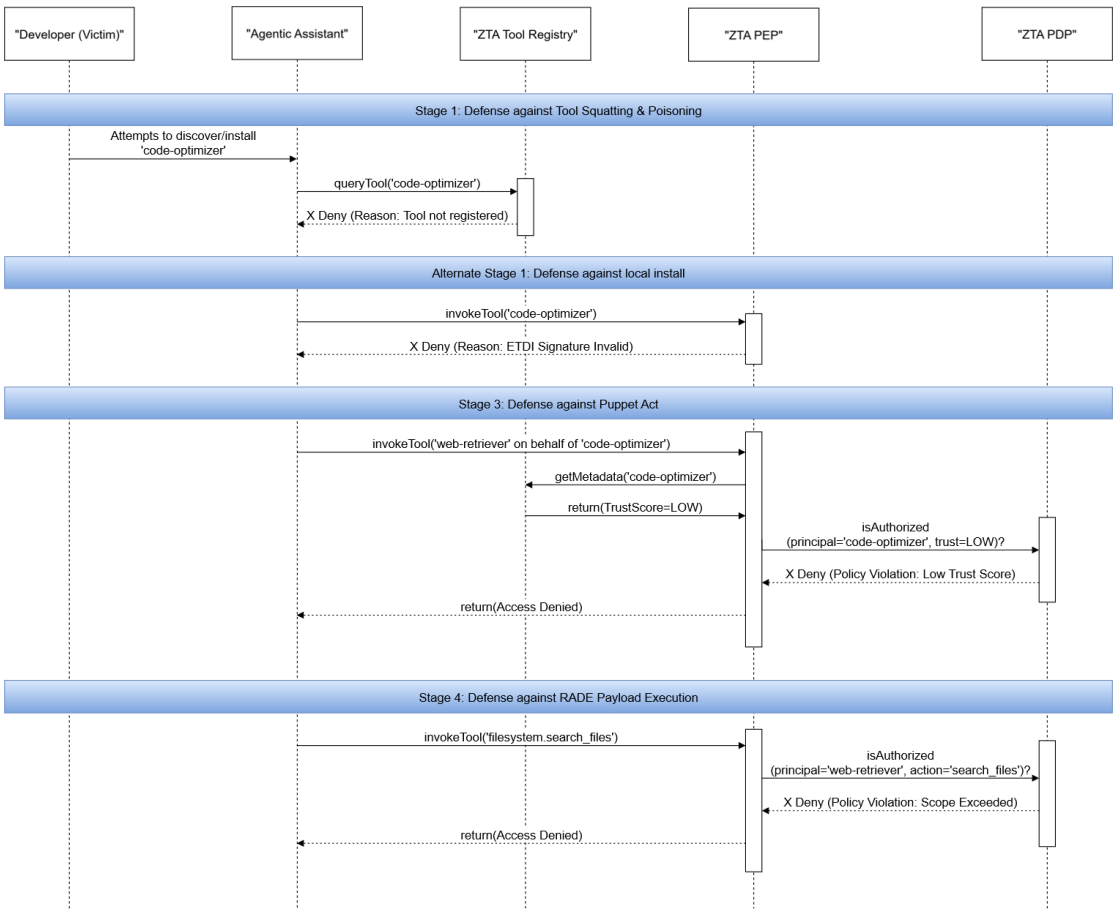


Figure 6 Puppet-RADE Attack & Defence Flow

This sequence diagram visualising the multi-stage Puppet-RADE attack scenario. It demonstrates the defensive interventions of the proposed architecture's components (Tool Registry, PEP, PDP) at each stage of the attack, showing how the chain is broken. The PDP would deny both requests, the actions would be logged as policy violations, and a high-priority security alert would be triggered, rendering the RADE payload inert and completely neutralising the

attack. The complete sequence of this multi-stage attack and the corresponding defensive interventions at each step are visualised in *Figure 6*.

The preceding analysis has demonstrated through simulated attack scenarios that the MCP ecosystem is inherently vulnerable to sophisticated, deception-based threats. The successful execution of "Tool Poisoning" and the subsequent unauthorised data access confirms that ad-hoc security controls are insufficient to protect agentic AI systems. These findings underscore the urgent need for a comprehensive, architecture-driven approach to security that is both robust and systematically designed. To construct such a solution, the analysis will now employ the SABSA framework, a business-driven enterprise architecture methodology, to translate the high-level principles of Zero Trust into a concrete and traceable security design. The following sections will detail this process, applying the six layers of the SABSA model to build a security architecture that directly mitigates the identified threats.

4.3 Analysis: Designing a SABSA-Driven Zero Trust Architecture

This section details the primary research artifact of this dissertation: the comprehensive, multi-layered security architecture for the MCP ecosystem. The architecture is designed using the SABSA framework as a structured methodology and is grounded in the philosophy of Zero Trust. This approach provides a clear and traceable path from high-level business risks down to the specific technical components and standards required for implementation, directly addressing the threats demonstrated in the preceding adversarial simulation.

4.3.1 The SABSA Matrix: A Blueprint for Traceable Security

The Sherwood Applied Business Security Architecture (SABSA) is a holistic, risk-driven framework for developing enterprise security architecture that ensures all security decisions are traceable back to business requirements. Its purpose is to bridge the gap between high-level business goals and the technical implementation of security controls. The core of SABSA is the "SABSA Matrix," a 6x6 grid that provides a comprehensive structure for architectural analysis. It consists of six horizontal layers of abstraction that represent different perspectives on the system, from the high-level business context down to the specific technical components. These layers are intersected by six vertical columns that represent the fundamental interrogatives: What (Assets), Why (Motivation), How (Process), Who (People), Where (Location), and When (Time). This is shown clearly in the *Figure 7*. By systematically working through the matrix from the top down, this methodology ensures that the resulting

security architecture is a coherent system that directly supports business objectives. *Table 5* presents the completed SABSA Matrix for the proposed Zero Trust MCP ecosystem, which serves as a concise and comprehensive map of the entire security design.

	ASSETS (What)	MOTIVATION (Why)	PROCESS (How)	PEOPLE (Who)	LOCATION (Where)	TIME (When)
CONTEXTUAL ARCHITECURE	Business Decisions	Business Risk	Business Processes	Business Governance	Business Geography	Business Time Dependence
	Taxonomy of Business Assets, including Goals & Objectives	Opportunities & Threats Inventory	Inventory of Operational Processes	Organisational Structure & the Extended Enterprise	Inventory of Buildings, Sites, Territories, Jurisdictions, etc.	Time dependencies of business objectives
CONCEPTUAL ARCHITECTURE	Business Knowledge & Risk Strategy	Risk Management Objectives	Strategies for Process Assurance	Roles & Responsibilities	Domain Framework	Time Management Framework
	Business Attributes Profile	Enablement & Control Objectives; Policy Architecture	Process Mapping Framework; Architectural Strategies for ICT	Owners, Custodians and Users; Service Providers & Customers	Security Domain Concepts & Framework	Through-Life Risk Management Framework
LOGICAL ARCHITECTURE	Information Assets	Risk Management Policies	Process Maps & Services	Entity & Trust Framework	Domain Maps	Calendar & Timetable
	Inventory of Information Assets	Domain Policies	Information Flows; Functional Transformations; Service Oriented Architecture	Entity Schema; Trust Models; Privilege Profiles	Domain Definitions; Inter-domain associations & interactions	Start Times, Lifetimes & Deadlines
PHYSICAL ARCHITECTURE	Data Assets	Risk Management Practices	Process Mechanisms	Human Interface	ICT Infrastructure	Processing Schedule
	Data Dictionary & Data Inventory	Risk Management Rules & Procedures	Applications; Middleware; Systems; Security Mechanisms	User Interface to ICT Systems; Access Control Systems	Host Platforms, Layout & Networks	Timing & Sequencing of Processes and Sessions
COMPONENT ARCHITECTURE	ICT Components	Risk Management Tools & Standards	Process Tools & Standards	Personnel Man'ment Tools & Standards	Locator Tools & Standards	Step Timing & Sequencing Tools
	ICT Products, including Data Repositories and Processors	Risk Analysis Tools; Risk Registers; Risk Monitoring and Reporting Tools	Tools and Protocols for Process Delivery	Identities; Job Descriptions; Roles; Functions; Actions & Access Control Lists	Nodes, Addresses and other Locators	Time Schedules; Clocks, Timers & Interrupts
SERVICE MANAGEMENT ARCHITECTURE	Service Delivery Management	Operational Risk Management	Process Delivery Management	Personnel Management	Management of Environment	Time & Performance Management
	Assurance of Operational Continuity & Excellence	Risk Assessment; Risk Monitoring & Reporting; Risk Treatment	Management & Support of Systems, Applications & Services	Account Provisioning; User Support Management	Management of Buildings, Sites, Platforms & Networks	Management of Calendar and Timetable

Figure 7 SABSA Matrix

Table 5 The SABSA Matrix for a Zero Trust MCP Ecosystem

	ASSETS (What)	MOTIVATION (Why)	PROCESS (How)	PEOPLE (Who)	LOCATION (Where)	TIME (When)
CONTEXTUAL (Business View)	Business Assets: Integrity of AI-driven processes, confidentiality of user/corporate data, availability of AI services, brand reputation.	Business Risk: Mitigate data exfiltration, financial fraud, and system compromise. Enable secure AI innovation and ensure regulatory compliance.	Business Processes: Automated agentic workflows (e.g., financial analysis, software development, customer service) using MCP tools.	Business Governance: End-users, AI application developers, tool providers, enterprise security and governance teams.	Business Geography: Distributed environments including on-premise data centres, multi-cloud platforms, and end-user devices.	Business Time-Dependence: Real-time, continuous (24/7) security enforcement to support uninterrupted business operations.
CONCEPTUAL (Architect's View)	Business Attributes Profile: Verifiability, Integrity, Confidentiality, Authorisation, Non-repudiation, Auditability.	Risk Management Objectives: Enforce "never trust, always verify" (ZTA). Ensure verifiable identity for all entities and mandate explicit, dynamic Authorisation per-session.	Strategies for Process Assurance: Implement a Defence-in-depth security model that integrates identity, device, network, application, and data pillars of ZTA.	Roles & Responsibilities: Security architects define principles. Asset owners define risk appetite. Custodians implement controls.	Domain Framework: A universal security domain covering the entire MCP ecosystem, where no entity is implicitly trusted.	Time Management Framework: A through-life risk management approach, applying security from initial design through to decommissioning.
LOGICAL (Designer's View)	Information Assets: Tool definitions, user credentials, agent identities, API keys, sensitive data in transit.	Risk Management Policies: Policies mandating cryptographic verification of all tool definitions and dynamic, context-aware Authorisation for all tool invocations.	Process Maps & Services: Logical security services: Identity Management, Tool Registry, Tool Definition Integrity, PAP, PDP, PEP, Secure Communications, and Auditing.	Entity & Trust Framework: Abstract entities (principals, resources, actions) and a trust model based on verifiable attributes, not network location.	Domain Maps: Logical segmentation of services (e.g., identity, policy, enforcement) with defined trust boundaries and secure interaction protocols.	Calendar & Timetable: Services must be continuously available to make real-time Authorisation decisions for every tool invocation.

PHYSICAL (Builder's View)	Data Assets: Signed tool definitions (ETDI), JWTs, policy files (Rego/Cedar), PKI certificates, SIEM logs.	Risk Management Rules: Policies implemented in a policy engine (e.g., OPA) that deny access by default unless explicitly permitted by a rule.	Process Mechanisms: API Gateway (PEP) intercepts requests. Policy Engine (PE) evaluates policies. OAuth Server issues tokens. Tool Registry provides discovery.	Human Interface: Administrator-controlled portal for tool registration. User-facing consent prompts for new tool versions or scope changes.	ICT Infrastructure: Deployment of PEP as an inline proxy; PDP and Tool Registry as centralised microservices in a secure cloud environment.	Processing Schedule: Per-session, just-in-time credential issuance and per-request Authorisation checks, adding minimal latency to invocations.
COMPONENT (Tradesman's View)	ICT Components: JWS for signing tool definitions. JWTs (RFC 7519) for access tokens. TLS 1.3 for transport encryption.	Risk Management Tools: Automated scanners (e.g., McpSafetyScanner) integrated into the tool vetting process. SIEM for log analysis and alerting.	Process Tools & Standards: Rego or Cedar policy language. OAuth 2.0 framework for delegated Authorisation. "*OpenTelemetry*" for distributed tracing.	Personnel Management Tools: IAM platforms for managing user and service identities. Role-Based Access Control (RBAC) for administrators.	Locator Tools & Standards: mTLS for secure, authenticated service-to-service communication between PEP, PDP, and other backend components.	Step Timing & Sequencing Tools: Short-lived tokens with defined expiration times. Time-based conditions within Authorisation policies (e.g., time-of-day access).
SERVICE MGT. (Manager's View)	Service Delivery Management: Ensure continuous availability and integrity of security services. Monitor for configuration drift.	Operational Risk Management: Continuous monitoring for anomalies. Threat intelligence integration to update detection rules and policies.	Process Delivery Management: Formal tool vetting and lifecycle management. "Policy-as-code" CI/CD pipeline for managing Authorisation policies.	Personnel Management: Security Operations Centre (SOC) team managing monitoring and response. Administrators managing the Tool Registry.	Management of Environment: Secure management of the cloud environment hosting the security services. Secure key and certificate lifecycle management.	Time & Performance Management: Tailored incident response playbooks for MCP threats. Regular, automated access reviews and tool recertification.

This matrix serves as the central organising construct for the proposed architecture. It maps the six SABSA layers (rows) against the six fundamental interrogatives (columns) to ensure every security design decision is comprehensive and directly traceable to a business or risk driver. This matrix provides a comprehensive, top-down map of the security design. To illustrate how these layers translate into a functional system, *Figure 8* presents a logical diagram of the proposed ZTA Control Plane**.** This diagram shows the data flow for a typical tool invocation request, from its interception by the Policy Enforcement Point (PEP) to the dynamic Authorisation process involving the various security services detailed in the subsequent layers.

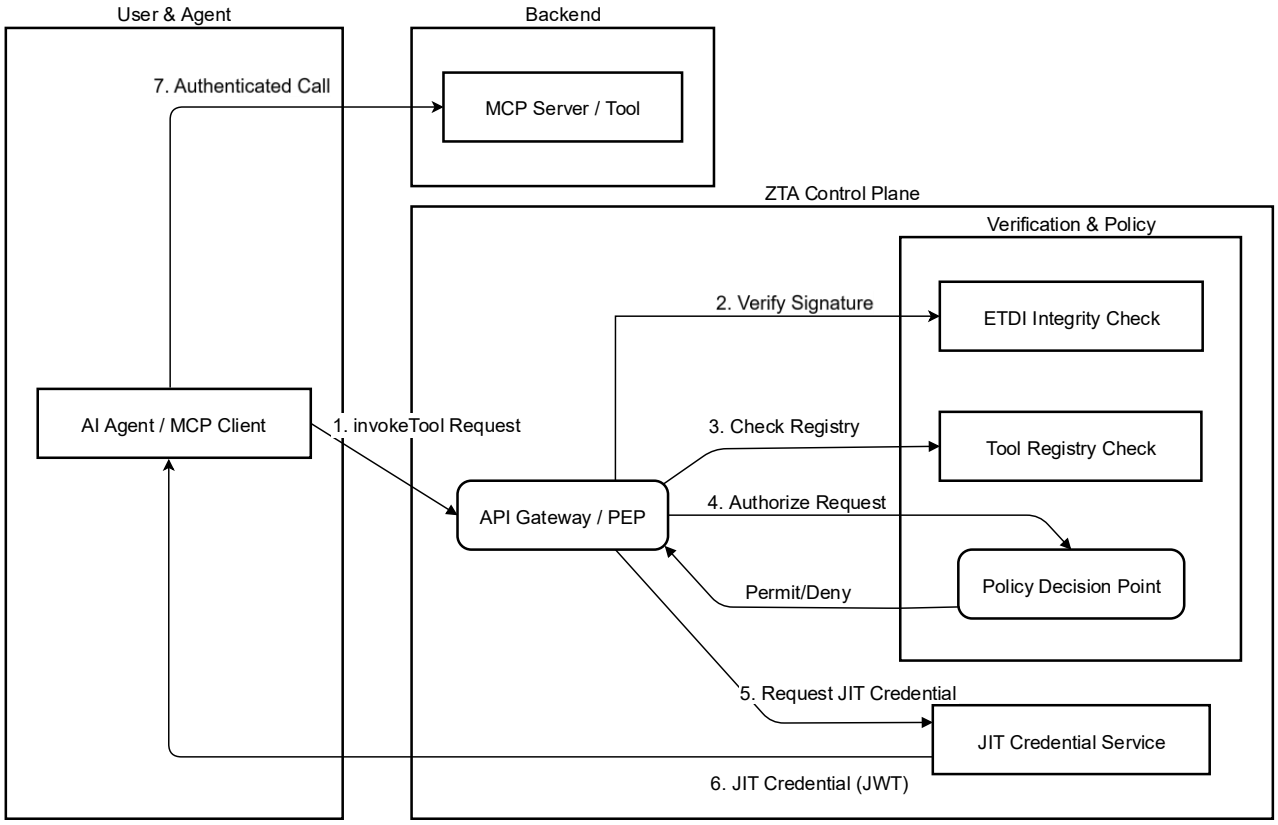


Figure 8 Proposed ZTA Architecture

This logical architecture diagram shows the proposed ZTA Control Plane. It illustrates the data flow of an agent's request, from interception by the PEP to the dynamic decision-making process involving the ETDI Service, Tool Registry, and PDP, culminating in the issuance of a JIT credential.

The overall security architecture can be described in terms of SABSA as follows:

4.3.2 The Contextual Layer (Business View)

The Contextual Layer defines the business environment, drivers, and risks that shape the security architecture. The standard MCP specification operates on a model of implicit trust, lacking the primitives to verify the identity of tool providers or the integrity of their offerings. This creates a significant business risk, as the open and extensible nature of the ecosystem becomes a largely ungoverned attack surface. The primary business motivation for this security architecture is, therefore, to mitigate the tangible threats that arise from this trust deficit. These risks include data exfiltration, unauthorised system access, financial fraud, and a significant erosion of user trust in agentic AI technologies, all of which can have severe financial and reputational consequences for the enterprise. By embedding a verifiable trust framework into the MCP ecosystem, the architecture's purpose is to transform MCP from a high-risk protocol into a secure enabler of AI innovation, allowing the organisation to confidently leverage agentic AI for critical business processes while ensuring compliance with data protection regulations and maintaining stakeholder trust.

4.3.3 The Conceptual Layer (Architect's View)

The Conceptual Layer translates the business requirements from the Contextual Layer into a high-level security strategy and a set of core principles. This is where the philosophy of Zero Trust is formally adopted as the guiding architectural principle. To structure this translation, a "Business Attributes Profile" is defined using SABSA's methodology, translating business needs into a standardised set of security qualities the architecture must deliver. These attributes include "Verifiability," ensuring all entities have a cryptographically verifiable identity; "Integrity," protecting all tool definitions and communications from unauthorised modification; "Confidentiality," mandating encryption for all data in transit; "Authorisation," requiring that access to every tool and resource be explicitly and dynamically granted based on policy; "Non-repudiation," ensuring that actions are attributable; and "Auditability," requiring that all security-relevant events are logged for analysis. These attributes are then mapped to the seven core tenets of Zero Trust Architecture from NIST SP 800-207, which serve as the architecture's fundamental "Control Objectives". This mapping creates a clear and defensible link between a business need, such as preventing fraud, and the corresponding security principle, such as mandating dynamic Authorisation for all actions.

4.3.4 The Logical Layer (Designer's View)

The Logical Layer defines the technology-agnostic security services required to fulfil the conceptual control objectives. These are abstract building blocks that describe what security functions are needed, not how they will be implemented with specific products. The core logical services for this architecture include an "Identity & Credential Management Service" for validating all principals; a "Tool Registry & Secure Discovery Service" to provide a single, authoritative source for available tools; a "Tool

Definition Integrity Service" to guarantee authenticity; a "Policy Administration Point" (PAP) for managing security policies; a "Policy Decision Point" (PDP) to evaluate access requests against those policies; a "Policy Enforcement Point" (PEP) to intercept all requests and enforce PDP decisions; a "Secure Communications Service" to ensure confidentiality; and a "Security Monitoring & Auditing Service" to collect logs and provide visibility.

4.3.5 The Physical Layer (Builder's View)

The Physical Layer maps the abstract logical services to actual physical mechanisms and technologies, integrating the specific solutions identified in the literature review. In this architecture, the "Tool Definition Integrity Service" is physically implemented using the "Enhanced Tool Definition Interface" (ETDI), where tool definitions are digitally signed by their providers. The "Tool Registry & Secure Discovery Service" is implemented as a centralised, administrator-controlled database. The PDP is implemented using a dedicated policy engine like "Open Policy Agent" (OPA) or a managed service that uses the Cedar policy language. The PEP is implemented as an "API Gateway" or a dedicated middleware component, such as "MCP Guardian," which is placed in-line to intercept all tool invocation requests. The "Identity & Credential Management Service" is realised using an "OAuth 2.0 Authorisation Server" that issues short-lived, narrowly scoped access tokens, achieving the concept of "Just-in-Time" (JIT) credentials.

4.3.6 The Component Layer (Tradesman's View)

The Component Layer provides the final level of technical detail, specifying the individual components, protocols, and standards that comprise the physical mechanisms. For this architecture, digital signatures for ETDI are implemented using "JSON Web Signature" (JWS). Access tokens for JIT credentials are implemented as "JSON Web Tokens" (JWTs) as defined in RFC 7519, containing claims for the principal's identity and Authorised scopes. Secure communication is implemented using "TLS 1.3" for transport-layer encryption, with "mutual TLS" (mTLS) used for strong, certificate-based authentication between backend services like the PEP and PDP. Authorisation policies for the PDP are written in a declarative language such as "Rego" or "Cedar". Finally, all cryptographic keys are managed using a secure "Public Key Infrastructure" (PKI).

4.3.7 The Service Management Layer (Manager's View)

The Service Management Layer addresses the ongoing management, monitoring, and measurement of the security services throughout their lifecycle to ensure the architecture remains effective over time. This includes "Comprehensive Monitoring and Logging," where all security components send detailed logs to a central SIEM to enable dashboards and alerts for security-relevant events. It also requires "Tailored Incident Response," with specific playbooks developed for MCP-related events like a suspected Tool Poisoning attack. A "Formal Tool Vetting and Lifecycle Management" process is established for

onboarding, reviewing, and retiring tools, which includes automated security scanning. Finally, a "Policy-as-Code" approach is adopted for managing Authorisation policies, where policies are stored in version control and deployed through a secure CI/CD pipeline, ensuring changes are managed in a controlled and auditable manner.

4.4 Threat-to-Measure Traceability Matrix

To validate the completeness of the proposed architecture, it is essential to demonstrate a clear, traceable link between every identified threat and the specific security control designed to mitigate it. This ensures that the architecture is not merely a collection of security features but a cohesive system where each component serves a deliberate defensive purpose. *Table 6* provides this traceability, summarising the dissertation's core problem-solution mapping.

Table 6 Threat-to-Measure Traceability Matrix

Identified Threat	Primary Exploited Vulnerability	Mitigating Architectural Services & Controls
Tool Poisoning	Lack of tool definition integrity and authenticity.	Tool Definition Integrity Service (ETDI with JWS); Content Security Policy for tool descriptions; Input validation at the PEP.
Rug Pull Attacks	Mutability of server-side logic and lack of re-approval for changes.	Immutable & Versioned Definitions (ETDI); PDP/PE re-validating access on every version change.
Tool Squatting	Lack of a trusted, centrally-verified discovery mechanism.	Tool Registry & Secure Discovery Service (Administrator-controlled, authenticated access).
Puppet Attacks	Implicit trust in the agent's planning logic and ability to chain tools.	PDP/PEP with policies preventing low-trust principals from invoking other tools; Agent state isolation.
RADE Attacks	Implicit trust in data retrieved from external sources.	PEP/PDP enforcing strict least-privilege on tool actions (e.g., denying "*filesystem*" access to a retrieval tool); Output filtering and DLP.
Preference Manipulation	Manipulation of LLM selection via persuasive tool descriptions.	Tool Registry (suggests descriptions for manipulative language); PDP/PEP (policies can factor in tool reputation/trust scores).

This matrix provides a direct traceability link between the primary threats identified in the literature review and the specific mitigating architectural services and controls designed in this chapter. It serves as a concise summary of the architecture's defensive coverage.

4.5 Chapter Summary

This chapter presented the core findings and analysis of the research. It began with the results of a detailed adversarial simulation, walking through a sophisticated, multi-stage "Puppet-RADE" attack to demonstrate the critical security vulnerabilities in the standard MCP ecosystem and how the proposed architecture systematically mitigates them at each step. Following the results, the chapter provided the analysis by detailing the proposed solution: a comprehensive, SABSA-driven Zero Trust Architecture. This analysis outlined each of the six layers of the SABSA framework, from the high-level business context down to the specific technical components and operational management procedures, ensuring every architectural decision is traceable to a security requirement. This chapter has therefore presented both the problem in a practical context and the detailed architectural solution.

Chapter 5: Discussion

5.1 Introduction

This chapter synthesises the core elements of the dissertation. The objective is to connect the initial problem statement and literature review with the architectural solution developed in the preceding chapter. Section 5.2 begins by revisiting the key findings from the literature that established the research gap. Section 5.3 evaluates the effectiveness of the chosen research methodology in addressing this gap. A high-level summary of the dissertation's primary findings is presented in section 5.4, followed by an acknowledgment of the research limitations in section 5.5. Section 5.6 details the specific contributions of this work to both academia and industry. The chapter concludes with a final summary in section 5.7.

5.2 Key Findings from the Literature

The literature review confirmed that the technological landscape is undergoing a significant shift towards autonomous, agentic AI, with multi-agent systems (MAS) emerging as a powerful paradigm for solving complex problems. Central to this evolution is the Model Context Protocol (MCP), which has been rapidly adopted by industry leaders as a de facto standard to solve the problem of fragmented tool integration. The research established that while MCP is a critical enabler, its open, client-server architecture and lack of inherent security primitives create a new and poorly understood attack surface, operating on a fundamentally broken trust model.

This foundational vulnerability gives rise to a taxonomy of novel, AI-centric threats. The literature identified direct identity and integrity-based attacks such as "Tool Poisoning," where malicious instructions are hidden in tool descriptions, and "Rug Pull" attacks, a form of supply-chain compromise where a trusted tool’s functionality is altered post-approval. Further analysis revealed discovery-based threats like "Tool Squatting" and more sophisticated, chained attacks, including "Puppet Attacks," where a malicious tool manipulates trusted tools, and "Retrieval-Agent Deception" (RADE), where the agent is compromised by ingesting malicious external data.

In response to these threats, the literature review identified Zero Trust Architecture (ZTA) as the most appropriate security philosophy, grounded in the principle of "never trust, always verify". While the research community has proposed valuable but fragmented technical controls such as the Enhanced Tool Definition Interface (ETDI), middleware like MCP Guardian, and secure tool registries. All these acts as point solutions. The review therefore culminated in the identification of a critical research gap: the absence of a holistic, structured, and business-driven architectural methodology to systematically apply ZTA principles and integrate these disparate controls into a single, coherent security framework.

5.3 Effectiveness of Research Methodology

The quantitative modelling design chosen for this research proved highly effective in addressing the identified research gap. The methodology's strength lies in its systematic approach to constructing a conceptual model from an empirical analysis of existing, authoritative sources. By collecting data from protocol specifications, threat intelligence reports, and established security frameworks like those from the DoD and GSA, the methodology provided a solid, evidence-based foundation for the architectural design. The use of the SABSA framework as the analytical tool was particularly effective, as it provided a structured, multi-layered approach to translate the abstract principles of Zero Trust into a concrete set of implementable security controls. This design was instrumental in ensuring that every component of the proposed architecture is directly traceable to a specific business requirement or security threat, thereby successfully answering the core research question.

5.4 Summary of Findings

The primary finding of this dissertation is the successful design of a comprehensive, SABSA-driven Zero Trust Architecture for the Model Context Protocol, which directly addresses the research gap identified in the literature. The research demonstrated that by employing the SABSA framework's structured, multi-layered methodology, the abstract principles of Zero Trust can be systematically translated into a concrete and traceable security design for the MCP ecosystem. This business-driven approach ensures that every architectural component and security control, from high-level policy down to technical implementation, is directly linked to mitigating specific business risks like data exfiltration and system compromise, as defined in the Contextual and Conceptual layers of the architecture.

The core of the proposed solution is a ZTA Control Plane built upon the logical components defined by authoritative sources like the DoD and GSA: a "Policy Enforcement Point" (PEP), a "Policy Decision Point" (PDP), and a "Policy Administrator" (PA). In the proposed architecture, the PEP is realised as an API Gateway that intercepts all agent-tool communications. The PDP is implemented with a dedicated policy engine, such as one using the Cedar language, to evaluate every request against fine-grained, context-aware policies. This core is supported by other critical logical services, including a centralised "Tool Registry" to prevent squatting and ensure secure discovery, an "ETDI Service" to cryptographically verify tool identity and integrity, and a "Credential Management Service" to issue just-in-time (JIT) access tokens.

The dissertation validated the effectiveness of this architecture through a multi-stage adversarial simulation scenario that chained together the "Tool Poisoning," "RADE," and "Puppet Attack" vectors. The analysis demonstrated how the architecture's layered defences would systematically neutralise this sophisticated attack at each stage. For instance, the "Tool Registry" and ETDI's signature verification prevent the initial foothold from "Tool Squatting" and "Poisoning," while the PDP’s granular Authorisation

policies, enforcing the principle of least privilege, block the "Puppet Attack" and the final data exfiltration by denying trusted tools permission to perform anomalous actions. The findings confirm that this SABSA-driven ZTA successfully closes the security gaps in the standard protocol, providing a governable, enterprise-ready framework for agentic AI.

5.5 Research Limitations

This research is subject to several limitations. First, the proposed architecture is a conceptual model. While it is based on established frameworks and proven security principles, it has not been implemented or tested in a live, production enterprise environment. Its real-world performance and resilience would be subject to numerous variables not fully captured in a theoretical design. Second, the data collection relied on secondary sources. The analysis of threats is based on published research and threat models rather than primary data from active security incidents, which are often proprietary. Finally, the field of agentic AI and MCP is evolving at an exceptionally rapid pace. The threat landscape and available technologies are likely to change, and any static architectural blueprint will require continuous adaptation to remain effective over time.

5.6 Research Contribution

This dissertation offers a significant and timely contribution to the field of cybersecurity architecture by addressing the emergent security challenges of agentic AI. As organisations increasingly look to adopt LLM-powered agents and the MCP, they face a landscape of novel risks with little architectural guidance. This research bridges the critical gap between the high-level philosophy of Zero Trust and its practical, enterprise-grade application to secure these new systems.

The key contribution of this work is a practical, business-driven, and traceable security blueprint for the secure adoption and governance of agentic AI. It provides a novel application of the SABSA framework to architect a ZTA for MCP, demonstrating how established enterprise architecture methodologies can be effectively applied to mitigate the new threats posed by emerging technological paradigms, transforming MCP from a high-risk protocol into a secure enabler of AI innovation. For academics, it presents a new synthesis of enterprise architecture and AI security. For practitioners, it offers an actionable reference model for designing and implementing controls, enabling them to harness the power of agentic AI while managing its inherent risks.

5.7 Summary

In summary, this chapter has contextualised and synthesised the findings of the dissertation. It revisited the key conclusions from the literature review that motivated the research and affirmed the effectiveness of the chosen methodology in constructing the proposed architectural solution. A high-level overview of the findings was presented, followed by a transparent discussion of the research's limitations. Finally, the chapter articulated the work's primary contributions to both academic knowledge and professional practice, establishing the significance of applying formal architectural methods to secure emerging AI technologies.

Chapter 6: Conclusion

6.1 Introduction

This concluding chapter consolidates the research journey undertaken in this dissertation. Section 6.2 provides a comprehensive summary of the dissertation, retracing the logical path from the initial research motivation through the literature review, methodology, and key findings. Section 6.3 then explicitly demonstrates how the research objectives were successfully achieved. Building on this, section 6.4 outlines several recommendations for future research that can extend the work presented here. The chapter finishes in section 6.5 with a personal reflection on the research process and the key insights gained, and is wrapped up with a final summary in section 6.6.

6.2 Summary of the Research

This dissertation addressed the critical and emergent security vulnerabilities introduced by the Model Context Protocol (MCP) in the rapidly expanding field of agentic AI. The research began by establishing that while MCP provides a vital standard for interoperability, its architecture creates a new attack surface vulnerable to threats such as "Tool Poisoning" and "Rug Pull Attacks," which existing security paradigms are ill-equipped to handle. The literature review confirmed a critical gap: while Zero Trust Architecture (ZTA) offers the appropriate security philosophy, no formal, business-driven methodology existed to architect a ZTA specifically for the MCP ecosystem.

To address this, a quantitative modelling design was employed. This methodology guided the systematic analysis of the MCP threat landscape and the core components of ZTA, using the multi-layered SABSA framework as an analytical tool to structure the design of a solution. The primary outcome of this research is a comprehensive, SABSA-driven Zero Trust Architecture for MCP. The "Results and Analysis" chapter demonstrated how this architecture's layered controls, built around the logical ZTA components of a Policy Engine, Policy Administrator, and Policy Enforcement Point, directly mitigate the identified threats in a traceable and governable manner. The discussion chapter further synthesised these findings, establishing the work's contribution as a practical and academically grounded blueprint for the secure governance of agentic AI.

6.3 Achievement of Research Objectives

The research successfully met all four of its established objectives. The first objective, to critically evaluate the security threats inherent in the current MCP architecture by synthesising existing academic and industry research, was achieved in the comprehensive literature review detailed in Chapter 2. This section provided a detailed taxonomy of threats, including Tool Poisoning, Rug Pull attacks, and advanced chained attacks like RADE, establishing a clear understanding of the problem domain.

The second objective, to analyse the core principles of Zero Trust Architecture as a guiding security philosophy and the methodology of the SABSA as a structured design framework, was also fulfilled in Chapter 2. This section detailed the foundational tenets of ZTA as defined by NIST and explained the business-driven, layered methodology of the SABSA framework, setting the stage for their application to the MCP ecosystem.

The third objective, to synthesise ZTA principles and MCP-specific technical controls using the SABSA framework to construct a holistic, traceable, and business-driven security architecture, was achieved in Chapter 4. This chapter presented the primary research artifact, the SABSA matrix, and detailed the design of the architecture through its six layers, demonstrating a systematic and justifiable construction process.

Finally, the fourth objective, to demonstrate how the proposed architecture provides systematic and robust mitigation for the identified MCP threats, was achieved through the adversarial attack simulation in Chapter 4. This section provided a practical validation of the architecture's defensive capabilities, showing how its integrated controls would neutralise a sophisticated attack at each stage, thereby closing the security gap in the current protocol.

6.4 Recommendations for Future Research

The conceptual framework established in this dissertation opens several avenues for future research. An immediate and valuable next step would be the development of a reference implementation or a proof-of-concept prototype of the proposed architecture. Such a project would allow for empirical testing of the controls, providing performance benchmarks and validating their effectiveness against live attack scenarios. Furthermore, future work could focus on creating a ZTA maturity model specifically for MCP, which would provide organisations with a phased roadmap for incrementally adopting the security controls outlined in this dissertation.

A parallel research avenue involves extending the architecture to incorporate automated policy generation and enforcement. This could involve using AI-driven analytics to dynamically adapt access policies based on real-time threat intelligence and agent behaviour, moving towards a more adaptive and resilient security posture. Longer-term research could also explore the integration of decentralised identity solutions, such as Decentralised Identifiers (DIDs) and Verifiable Credentials (VCs), to further enhance the security and trustworthiness of the agent-tool identity management process within the proposed Zero Trust framework.

6.5 Reflections

The journey of researching and writing this dissertation has been a profound learning experience, requiring me to navigate the intersection of established enterprise architecture methodologies and the rapidly evolving frontier of agentic AI. The initial challenge was bridging these two disparate fields, which required a deep dive into both the structured, process-oriented world of the SABSA framework and the dynamic, often chaotic, landscape of AI security vulnerabilities. This process underscored the necessity of a methodical approach to manage the complexity of new technological paradigms.

A key insight I gained was the enduring relevance of foundational architectural principles in taming this complexity. While the threats posed by MCP are novel, the effective solutions are rooted in the timeless security solutions of traceability, business alignment, and explicit trust verification. This realisation reinforced my core argument that a systematic, business-driven approach is not only applicable but essential for engineering security into the next generation of AI systems. It became clear that the "what" and "why" of security, as defined by the business, must always drive the "how" of the technical implementation.

Ultimately, this project highlighted for me the immense difficulty and critical importance of creating practical blueprint that translate high-level security philosophies into actionable, enterprise-grade solutions. Moving from the abstract concept of "Zero Trust" to a detailed, layered architecture with specific controls and data flows was a challenging but rewarding exercise. It solidified my understanding that effective cybersecurity architecture is not just about implementing technologies, but about building a coherent, defensible system where every component serves a purpose that is clearly traceable back to the risks it is designed to mitigate.

References

Acharya, D.B., Kuppan, K. and Divya, B. (2025) 'Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey', *IEEE Access*, 13, pp. 18912–18936. Available at: https://doi.org/10.1109/ACCESS.2025.3532853.

Anthropic (2024) *Introducing the Model Context Protocol*. Available at: https://www.anthropic.com/news/model-context-protocol (Accessed: 7 June 2025).

Babar, Z. (2024) 'Securing AI Systems with MITRE ATLAS', *Medium*, 29 December. Available at: https://medium.com/@zbabar/securing-ai-systems-with-mitre-atlas-89a263f38772 (Accessed: 25 June 2025).

Bhatt, M., Narajala, V.S. and Habler, I. (2025) 'ETDI: Mitigating Tool Squatting and Rug Pull Attacks in Model Context Protocol (MCP) by using OAuth-Enhanced Tool Definitions and Policy-Based Access Control'. arXiv. Available at: https://doi.org/10.48550/arXiv.2506.01333.

Dash, B. (2024) 'Zero-Trust Architecture (ZTA): Designing an AI-Powered Cloud Security Framework for LLMs' Black Box Problems', *Current Trends in Engineering Science (CTES)*, 4(2), pp. 1–5. Available at: https://doi.org/10.54026/CTES/1058.

Edwards, S. (2023) *What is the Mitre Att&ck Framework? | CrowdStrike*, *CrowdStrike.com*. Available at: https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/mitre-attack-framework/ (Accessed: 20 June 2025).

Ehtesham, A. *et al.* (2025) 'A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)'. arXiv. Available at: https://doi.org/10.48550/arXiv.2505.02279.

Fu, S. *et al.* (2023) 'Jut: A Framework for Just-in-Time Data Access'. arXiv. Available at: https://doi.org/10.48550/arXiv.2312.01304.

Google (2025) *What is Model Context Protocol (MCP)? A guide*, *Google Cloud*. Available at: https://cloud.google.com/discover/what-is-model-context-protocol (Accessed: 8 June 2025).

Halloran, J. (2025) 'MCP Safety Training: Learning to Refuse Falsely Benign MCP Exploits using Improved Preference Alignment'. arXiv. Available at: https://doi.org/10.48550/arXiv.2505.23634.

Hasan, M.M. *et al.* (2025) 'Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers'. arXiv. Available at: https://doi.org/10.48550/arXiv.2506.13538.

Hou, X. *et al.* (2025) 'Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions'. arXiv. Available at: https://doi.org/10.48550/arXiv.2503.23278.

Huang, K. *et al.* (2025) 'Agent Capability Negotiation and Binding Protocol (ACNBP)'. arXiv. Available at: https://doi.org/10.48550/arXiv.2506.13590.

Karim, M.M. *et al.* (2025) 'Transforming Data Annotation with AI Agents: A Review of Architectures, Reasoning, Applications, and Impact', *Future Internet*, 17(8), p. 353. Available at: https://doi.org/10.3390/fi17080353.

Kong, D. *et al.* (2025) 'A Survey of LLM-Driven AI Agent Communication: Protocols, Security Risks, and Defense Countermeasures'. arXiv. Available at: https://doi.org/10.48550/arXiv.2506.19676.

Krishnan, N. (2025) 'Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications'. arXiv. Available at: https://doi.org/10.48550/arXiv.2504.21030.

Kumar, S. *et al.* (2025) 'MCP Guardian: A Security-First Layer for Safeguarding MCP-Based AI System', in *Advanced Natural Language Processing 2025*. *6th International Conference on Advanced Natural Language Processing*, Academy & Industry Research Collaboration Center, pp. 107–121. Available at: https://doi.org/10.5121/csit.2025.150908.

Kumar, V. (2025) 'MITRE ATLAS Framework 2025 - Guide to Securing AI Systems', *Practical DevSecOps*, 10 July. Available at: https://www.practical-devsecops.com/mitre-atlas-framework-guide-securing-ai-systems/ (Accessed: 25 June 2025).

Li, Q. and Xie, Y. (2025) 'From Glue-Code to Protocols: A Critical Analysis of A2A and MCP Integration for Scalable Agent Systems'.

Liao, C.C., Liao, D. and Gadiraju, S.S. (2025) 'AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis'. arXiv. Available at: https://doi.org/10.48550/arXiv.2507.21105.

Lin, Z. *et al.* (2025) 'A Large-Scale Evolvable Dataset for Model Context Protocol Ecosystem and Security Analysis'. arXiv. Available at: https://doi.org/10.48550/arXiv.2506.23474.

Masood, A. (2025) 'Securing Large Language Models: A MITRE ATLAS Playbook', *Medium*, 25 February. Available at: https://medium.com/@adnanmasood/securing-large-language-models-a-mitre-atlas-playbook-5ed37e55111e (Accessed: 25 June 2025).

MCP (2025) *Architecture Overview*, *Model Context Protocol*. Available at: https://modelcontextprotocol.io/docs/learn/architecture (Accessed: 7 June 2025).

Microsoft (2025a) *Microsoft Learn MCP Server overview*. Available at: https://learn.microsoft.com/en-us/training/support/mcp (Accessed: 12 June 2025).

Microsoft (2025b) *What is the MITRE ATT&CK framework? | Microsoft Security*. Available at: https://www.microsoft.com/en-us/security/business/security-101/what-is-mitre-attack-framework (Accessed: 20 June 2025).

MITRE ATLAS (2025) *Tactics | MITRE ATLAS™*. Available at: https://atlas.mitre.org/tactics (Accessed: 2 July 2025).

Narajala, V.S., Huang, K. and Habler, I. (2025) 'Securing GenAI Multi-Agent Systems Against Tool Squatting: A Zero Trust Registry-Based Approach'. arXiv. Available at: https://doi.org/10.48550/arXiv.2504.19951.

OWASP (2024) *OWASP Top 10 for Large Language Model Applications | OWASP Foundation*. Available at: https://owasp.org/www-project-top-10-for-large-language-model-applications/ (Accessed: 22 June 2025).

Palo Alto Networks (2025) *What Is MITRE ATT&CK Framework?*, *Palo Alto Networks*. Available at: https://www.paloaltonetworks.com/cyberpedia/what-is-mitre-attack (Accessed: 20 June 2025).

Piyush Patil (2025) 'Inside the MCP Protocol: Revolutionizing data communication and system interoperability', *World Journal of Advanced Research and Reviews*, 26(1), pp. 3055–3071. Available at: https://doi.org/10.30574/wjarr.2025.26.1.1401.

Radosevich, B. and Halloran, J. (2025) 'MCP Safety Audit: LLMs with the Model Context Protocol Allow Major Security Exploits'. arXiv. Available at: https://doi.org/10.48550/arXiv.2504.03767.

Rotibi, A.O. *et al.* (2023) 'Extended Dependency Modeling Technique for Cyber Risk Identification in ICS', *IEEE Access*, 11, pp. 37229–37242. Available at: https://doi.org/10.1109/ACCESS.2023.3263671.

Sapkota, R., Roumeliotis, K.I. and Karkee, M. (2025) 'AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges', *Information Fusion*, 126, p. 103599. Available at: https://doi.org/10.1016/j.inffus.2025.103599.

Schneider, J. (2025) ‘Generative to Agentic AI: Survey, Conceptualization, and Challenges’. arXiv. Available at: https://doi.org/10.48550/arXiv.2504.18875.

Sedjelmaci, H. and Ansari, N. (2024) ‘Zero Trust Architecture Empowered Attack Detection Framework to Secure 6G Edge Computing’, *IEEE Network*, 38(1), pp. 196–202. Available at: https://doi.org/10.1109/MNET.131.2200513.

Sha, Z. *et al.* (2025) ‘Agent Safety Alignment via Reinforcement Learning’. arXiv. Available at: https://doi.org/10.48550/arXiv.2507.08270.

Shareef, S.H. (2025) ‘Model Context Protocol (MCP): A Universal Connector for AI and Data’, *Analytics Vidhya*, 25 February. Available at: https://www.analyticsvidhya.com/blog/2025/02/model-context-protocol/ (Accessed: 7 June 2025).

Sherwood, J., Clark, A. and Lynas, D. (2009) ‘Enterprise Security Architecture’.

Song, H. *et al.* (2025) ‘Beyond the Protocol: Unveiling Attack Vectors in the Model Context Protocol (MCP) Ecosystem’. arXiv. Available at: https://doi.org/10.48550/arXiv.2506.02040.

Song, W. *et al.* (2025) ‘Help or Hurdle? Rethinking Model Context Protocol-Augmented Large Language Models’. arXiv. Available at: https://doi.org/10.48550/arXiv.2508.12566.

Sowa, K. *et al.* (2025) ‘From Expert Systems to Generative Artificial Experts’.

Strom, B.E. *et al.* (2020) ‘MITRE ATT&CKÒ: Design and Philosophy’.

Syed, N.F. *et al.* (2022) ‘Zero Trust Architecture (ZTA): A Comprehensive Survey’, *IEEE Access*, 10, pp. 57143–57179. Available at: https://doi.org/10.1109/ACCESS.2022.3174679.

The Nightfall Team (2025) *MITRE ATLAS: The Essential Guide | Nightfall AI Security 101*. Available at: https://www.nightfall.ai/ai-security-101/mitre-atlas (Accessed: 26 June 2025).

Todd, W. (2025) *MCP is a powerful new AI coding technology: Understand the risks*, *ReversingLabs*. Available at: https://www.reversinglabs.com/blog/mcp-powerful-ai-coding-risk (Accessed: 7 June 2025).

U.S. Department of Defence (2022) *Zero Trust Reference Architecture (v2.0) - Defense Management Institute*. Available at: https://www.dmi-ida.org/knowledge-base-detail/Zero-Trust-Reference-Architecture-v2.0 (Accessed: 23 June 2025).

U.S. GSA (2025) ‘Zero Trust Architecture (ZTA)’.

Wang, F. (2025) ‘Model Context Protocol (MCP) real world use cases, adoptions and comparison to functional calling.’, *Medium*, 3 March. Available at: https://medium.com/@laowang_journey/model-context-protocol-mcp-real-world-use-cases-adoptions-and-comparison-to-functional-calling-9320b775845c (Accessed: 7 June 2025).

Wang, Z. *et al.* (2025) ‘MPMA: Preference Manipulation Attack Against Model Context Protocol’. arXiv. Available at: https://doi.org/10.48550/arXiv.2505.11154.

Xing, W. *et al.* (2025) ‘MCP-Guard: A Defense Framework for Model Context Protocol Integrity in Large Language Model Applications’. arXiv. Available at: https://doi.org/10.48550/arXiv.2508.10991.

Appendices

Appendix A: Ethical Training Certifications

Below are the figures that shows all the mandated ethical training were successfully completed.

Research Integrity Training

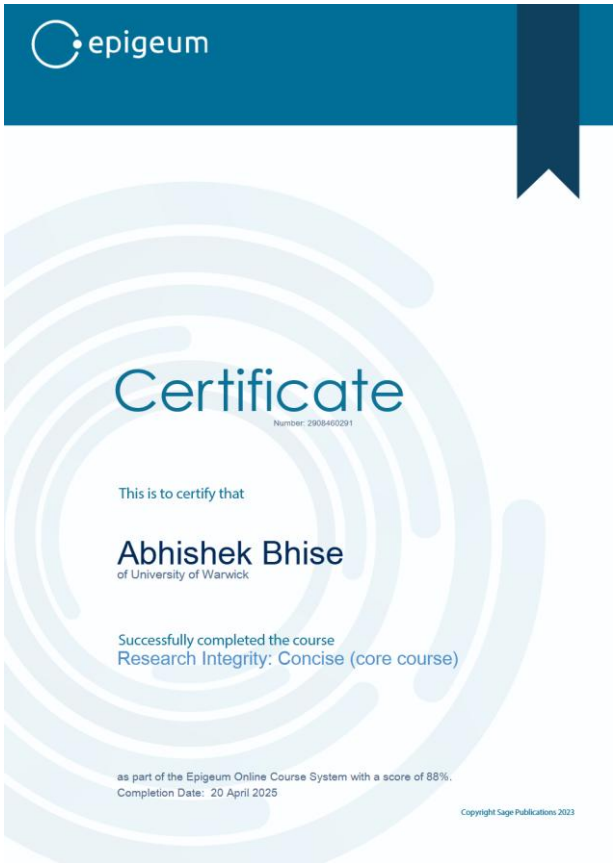


Figure A1 Research Integrity Training

Information System Smart Training

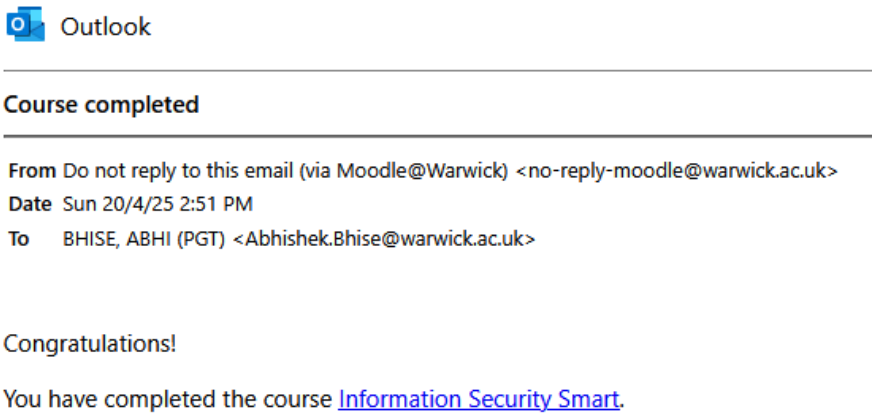


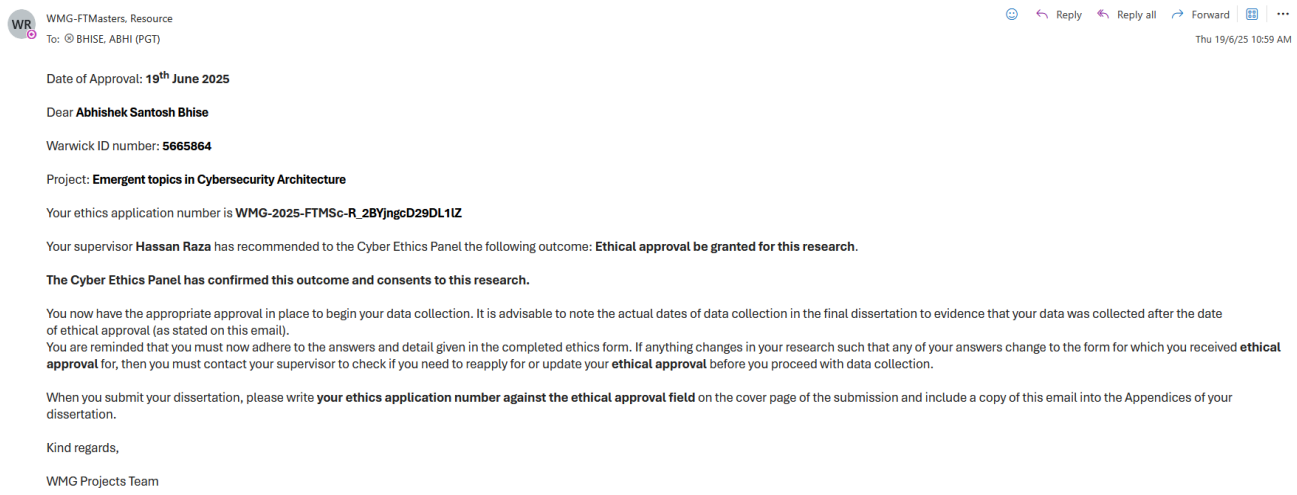
Figure A2 Information System Smart Training

WMG Student Ethics Training

Figure A3 WMG Student Ethics Training

Appendix B: Ethical Approval

This research was conducted in strict adherence to all the ethical guidelines. Prior to the commencement of the study, formal approval was obtained from the university's Cyber Ethics Panel. A copy of the official ethical approval is provided below.



WMG-FTMasters, Resource
To: BHISE, ABHI (PGT)

Reply Reply all Forward
Thu 19/6/25 10:59 AM

Date of Approval: **19th June 2025**

Dear **Abhishek Santosh Bhise**

Warwick ID number: **5665864**

Project: **Emergent topics in Cybersecurity Architecture**

Your ethics application number is **WMG-2025-FTMSc-R_2BYjngcD29DL1IZ**

Your supervisor **Hassan Raza** has recommended to the Cyber Ethics Panel the following outcome: **Ethical approval be granted for this research**.

The Cyber Ethics Panel has confirmed this outcome and consents to this research.

You now have the appropriate approval in place to begin your data collection. It is advisable to note the actual dates of data collection in the final dissertation to evidence that your data was collected after the date of ethical approval (as stated on this email).
You are reminded that you must now adhere to the answers and detail given in the completed ethics form. If anything changes in your research such that any of your answers change to the form for which you received **ethical approval** for, then you must contact your supervisor to check if you need to reapply for or update your **ethical approval** before you proceed with data collection.

When you submit your dissertation, please write **your ethics application number against the ethical approval field** on the cover page of the submission and include a copy of this email into the Appendices of your dissertation.

Kind regards,

WMG Projects Team

Figure B1 Ethical approval confirmation