

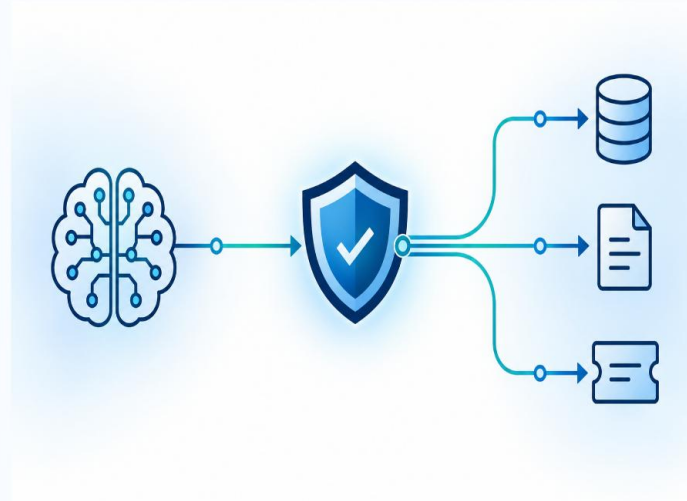
Dissertation Presentation

Adaptive Zero Trust Framework for Securing Autonomous AI Agents Against Unauthorized Tool Execution

MSc Dissertation Presentation

Md Riad Hossain | Student ID: 2434324
MSc Cyber Security & Digital Forensics
University of Wales Trinity Saint David — London
Campus
2026

Core idea: the AI agent may request an action,
but it should not automatically have authority to
execute it.



AI request → Zero Trust checks → controlled tool
access

My dissertation is about securing autonomous AI agents when they are connected to tools such as databases, files and support systems.

The main idea is very simple: an AI agent can understand a request and ask to use a tool, but that does not mean the action should be executed automatically.

My research places an independent Zero Trust security layer between the AI agent and the enterprise tools. This layer checks identity, permissions, risk and policy before an action can continue.

In this presentation I will explain the problem, the proposed framework, the Python prototype, the evaluation, the main findings, and the limitations.

2. Presentation Roadmap

A simple path through the dissertation

- 1 Problem** Why tool-enabled AI agents create a new security risk
- 2 Research** Aim, literature review, objectives and research gap
- 3 Design** Methodology and proposed Zero Trust framework
- 4 Prototype** How the Python system works
- 5 Evaluation** Test scenarios and what the prototype showed
- 6 Conclusion** Contribution, limitations and future work

I will keep the presentation in six simple parts. First I will explain the problem and why an AI agent with tool access creates a different security risk from a normal chatbot.

Then I will explain the research aim, the gap I identified, and the research methodology.

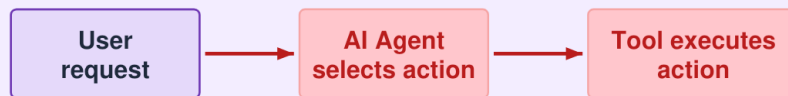
After that I will show the proposed Zero Trust framework and the Python prototype. I will then explain how I evaluated the prototype using different users, permissions and actions.

Finally, I will discuss what the research contributes, its limitations, and how it can be improved in future work.

3. The Security Problem

A tool-enabled AI agent can turn a wrong decision into a real action

Without an independent control



If the agent is manipulated, misunderstands the request, or selects a sensitive action, the effect can reach real data or systems.

Research problem: there is a need for an independent decision at the AI-agent-to-tool boundary. In simple terms, the system must check the requested action before execution instead of trusting the AI Agent automatically.

Main risks considered

- Prompt injection / agent hijacking
- Unauthorised tool execution
- Privilege misuse
- Sensitive data access
- File deletion or other high-impact actions

The problem starts when an AI agent can do more than generate text. A normal chatbot may give a wrong answer, but a tool-enabled agent can use a database, delete a file, create a ticket or access a report.

That means a wrong or manipulated decision can become a real system action. Prompt injection or agent hijacking can influence the agent, but even without an attack, the agent may misunderstand a request or select a sensitive operation.

Traditional authentication is still important, but it is not enough. A known user may still request an action that is outside their role or too risky.

So the core problem in my dissertation is the lack of an independent security decision between the AI agent and the tool it wants to use.

4. Research Aim and Objectives

What I wanted the research to achieve

Aim: design, implement and evaluate an Adaptive Zero Trust Framework that checks identity, permission, risk and policy before an AI-agent tool action is executed. The goal is to separate what the AI Agent wants to do from what it is actually allowed to do.

1. Investigate

Security threats around autonomous AI agents and tool execution

2. Examine

How Zero Trust principles can be applied to AI-agent security

3. Design

A framework using identity, permissions, risk and policy

4. Implement

A working Python prototype with enterprise-style tools

5. Evaluate

Different users, roles, actions, permissions and risk levels

The aim of the study was to design, implement and evaluate an Adaptive Zero Trust Framework for AI-agent tool execution.

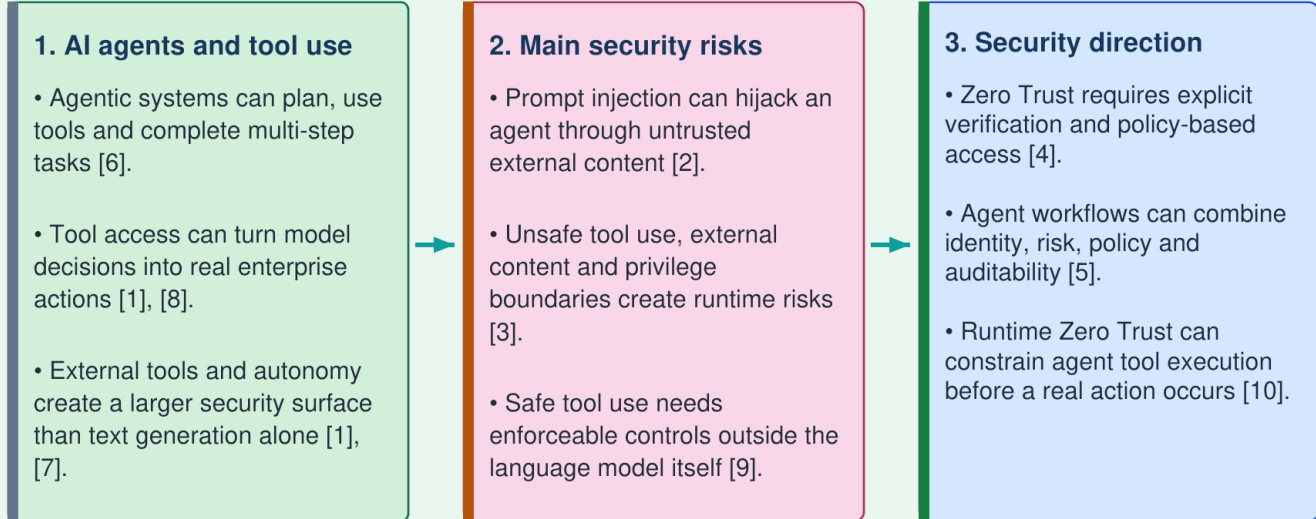
I divided this into five objectives. First, I reviewed the main security threats, especially unauthorised tool execution. Second, I examined Zero Trust ideas such as explicit verification and least privilege.

Third, I designed the framework. Fourth, I implemented it as a Python prototype with an AI Agent, identity and RBAC, risk engine, policy engine, decision engine, logger and tool manager.

Finally, I evaluated the framework using different users, requested actions, permissions and predefined risk levels. These objectives keep the work focused on runtime security around tool execution.

5. Literature Review Summary

What the literature says — and how it leads to my research gap



Research gap: the literature already discusses AI-agent risks, Zero Trust and safe tool use, but these controls are often studied separately. My dissertation brings identity, permission, risk, policy, logging and enforcement together in one request-by-request layer at the AI-agent-to-tool boundary.

This slide summarises the literature review in three simple parts.

First, previous research explains why agentic AI creates a new security problem. Agents can plan, use tools and complete multi-step tasks, so a wrong or manipulated decision can become a real action [1], [6], [7], [8].

Second, the literature identifies the main risks. NIST describes agent hijacking as a form of indirect prompt injection [2]. Other studies show that unsafe tool use, external content and privilege boundaries are important runtime risks [3], and safe tool use needs enforceable controls outside the model [9].

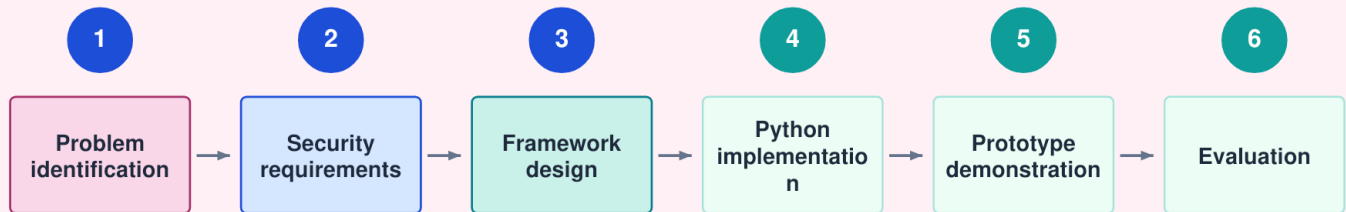
Third, the literature gives the security direction. NIST Zero Trust requires explicit verification and policy-based access [4]. Research on autonomous agents also combines identity, risk, policy and auditability [5], while runtime Zero Trust approaches aim to constrain tool execution [10].

So my gap is not that these ideas do not exist. My research brings them together in one simple request-by-request security layer at the AI-agent-to-tool boundary.

A simple way to explain this slide is: previous research tells us what the risks are and what controls are useful. My dissertation does not claim these ideas are new. The contribution is to combine them into one clear enforcement workflow and test that workflow in a Python prototype.

6. Research Methodology: Design Science Research

Build an artefact, demonstrate it, then evaluate it



Why this method fits

- The research produces a working cybersecurity artefact, not only a theoretical discussion. This means I could build the framework, run it, observe the decisions and evaluate whether the security controls behaved as expected.
- The prototype can be tested under controlled users, permissions, actions and risk levels.
- The evaluation shows how the proposed security design behaves in practice.

I used Design Science Research because the dissertation produces a working cybersecurity artefact: the Adaptive Zero Trust Framework and its Python prototype.

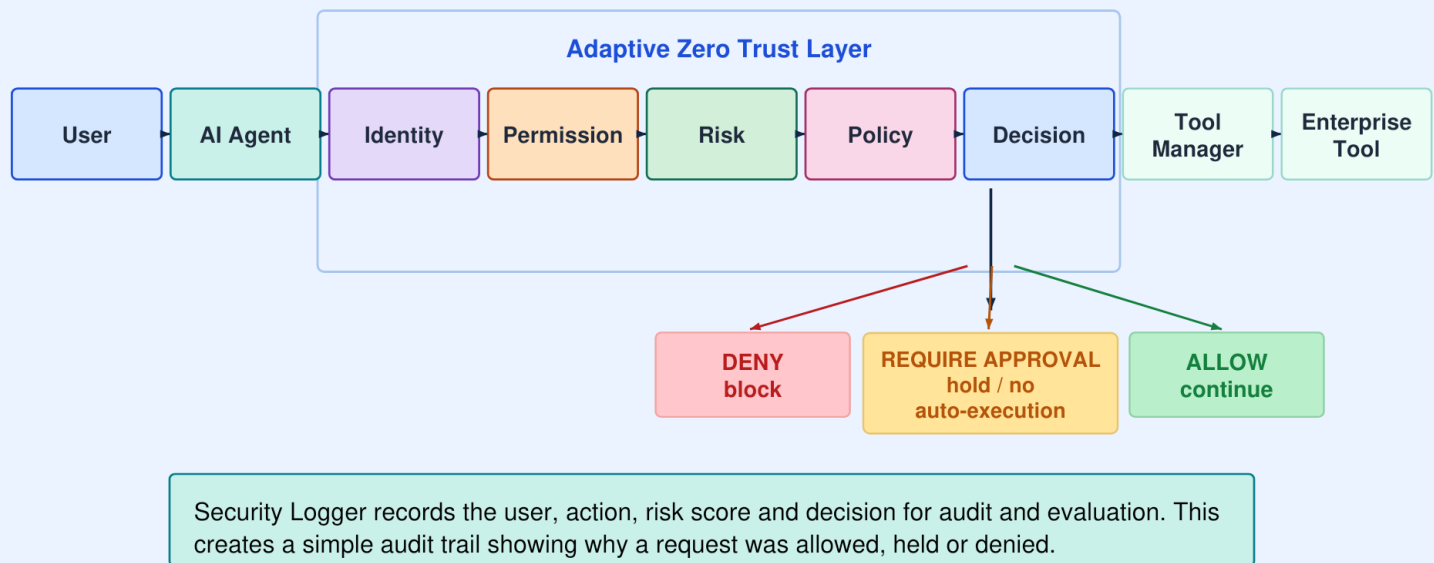
The process starts with problem identification from the literature. Then I define the security requirements and design the framework. After that I implement the design in Python and demonstrate it using enterprise-style actions.

Finally, I evaluate the prototype using controlled scenarios.

This methodology fits the project because I am not only discussing a security idea. I am building something that can be observed and tested. The evaluation then gives evidence about how the artefact behaves within the limits of the prototype.

7. Proposed Adaptive Zero Trust Framework

The AI Agent requests; the Zero Trust layer decides; the Tool Manager enforces



This is the core architecture of my dissertation. The user sends a natural-language request to the AI Agent. The agent identifies the requested action, but it does not execute the tool directly.

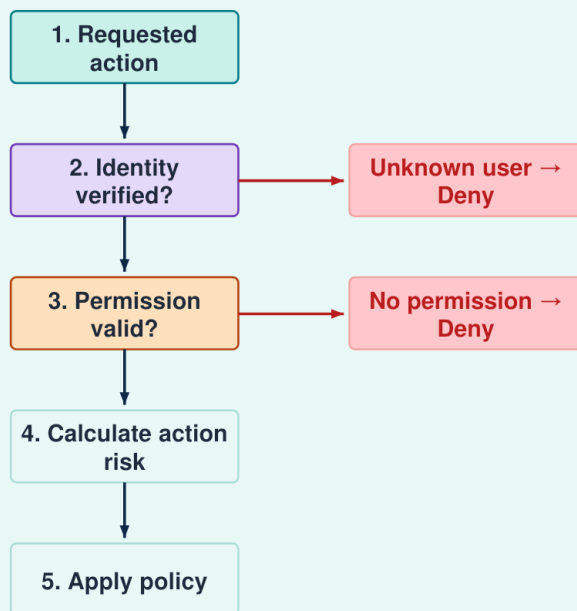
The request enters the Adaptive Zero Trust Layer. First, the user identity is checked. Then the system checks whether that user has permission for the requested action. If those checks pass, the Risk Engine assigns a predefined action risk and the Policy Engine applies the relevant policy.

The Decision Engine returns the outcome. The framework supports Allow, Require Approval and Deny. The Tool Manager is the final enforcement point.

The Security Logger records the request and decision. The most important design principle is that AI-generated intent is not the same as execution authority.

8. Security Decision Flow

Every request follows the same sequence of checks



Possible outcomes

ALLOW → Tool Manager may execute

REQUIRE APPROVAL → hold the request

DENY → block tool execution

All important events are logged. Only a request that passes the required security checks should reach the Tool Manager for execution.

The decision flow is intentionally simple and easy to follow. First, the AI Agent identifies the requested action. Then the Zero Trust system checks identity.

If the username is unknown, the request is denied immediately. If the user is recognised, the system checks whether the action exists in that user's permission list. A permission failure is also denied before the normal risk stage.

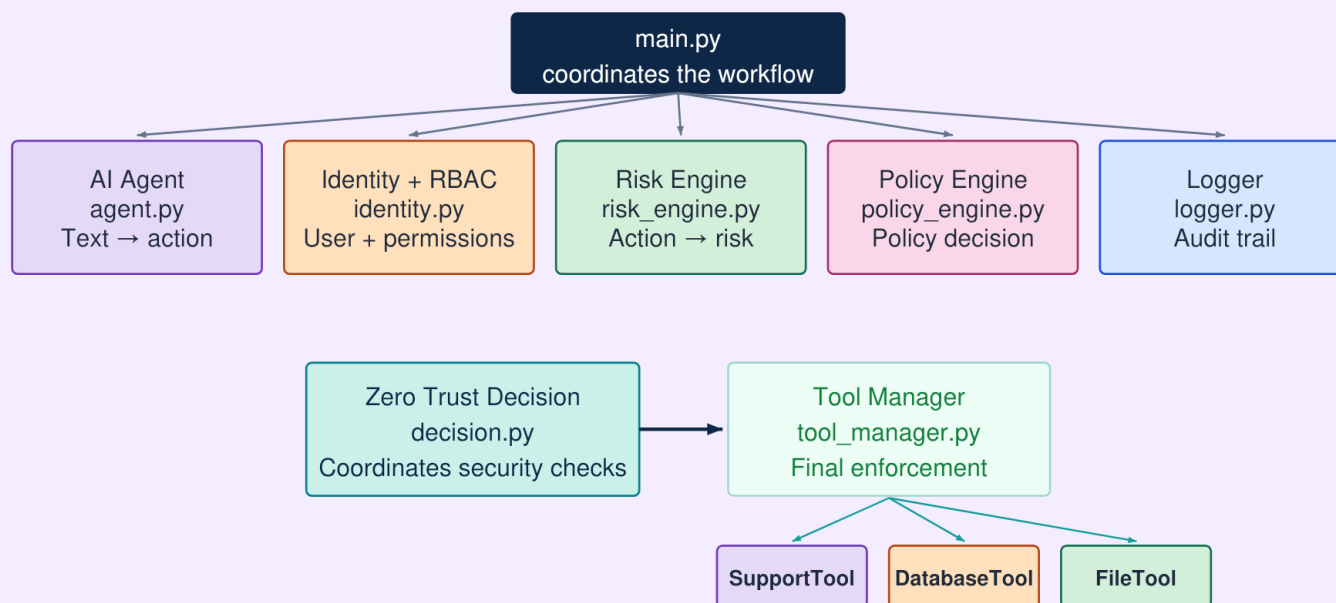
Only requests that pass identity and permission checks continue to the Risk Engine and Policy Engine.

The policy can produce Allow, Require Approval or Deny. In the framework, Require Approval means the operation should not execute automatically, although the current research does not implement a complete human approval workflow.

Important events are recorded by the Security Logger.

9. Python Prototype Structure

The implementation is modular so each security function is separate



The prototype is written in Python and divided into separate modules. The main application coordinates the workflow.

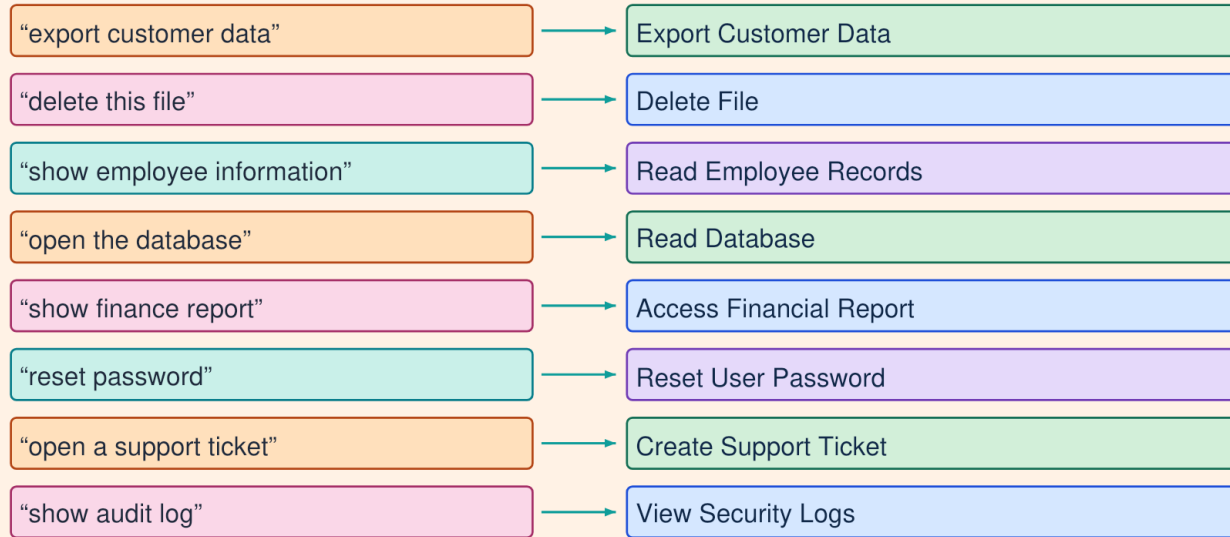
The AI Agent converts the user's natural-language request into a predefined action. The identity module stores the test users, roles and permission lists. The Risk Engine assigns the action risk. The Policy Engine produces the security outcome. The logger records events.

The Zero Trust Decision class coordinates these security checks. After a decision is produced, the Tool Manager acts as the enforcement point and routes permitted operations to the simulated Support, Database or File tools.

This modular design makes the prototype easier to understand and also means that simple components could later be replaced with more advanced enterprise services.

10. How the AI Agent Interprets Requests

The prototype uses keyword and phrase matching, not a production LLM planner



The AI Agent in this dissertation is deliberately simple. It is a Python class that checks keywords and phrases and maps the request to a predefined action.

For example, a request containing customer or client together with export, download or extract is mapped to Export Customer Data. A request about deleting or removing a file is mapped to Delete File.

Other mappings cover employee records, database access, financial reports, password reset, support tickets and security logs.

This is not a full LLM planner. That is a limitation, but it creates a controlled environment for testing the security layer.

Most importantly, the agent only identifies what action appears to be requested. Authorisation happens later in the Zero Trust system.

11. Identity and Role-Based Permissions

A recognised user can still be denied if the action is outside the assigned role

User	Role	Main permissions
admin	Administrator	Support ticket; employee records; security logs; financial reports; customer export; delete file; reset password
employee	Support employee	Create support ticket; read employee records
data_manager	Data manager	Read employee records; access financial report; export customer data

employee + Create Support Ticket →
permission exists

employee + Export Customer Data →
permission fails
This shows that a valid identity is not the
same as permission for every action.

The identity component contains three predefined users for the prototype: admin, employee and data_manager.

The administrator has the broadest permissions. The support employee has only routine permissions, such as creating a support ticket and reading employee records. The data manager has selected data-related permissions.

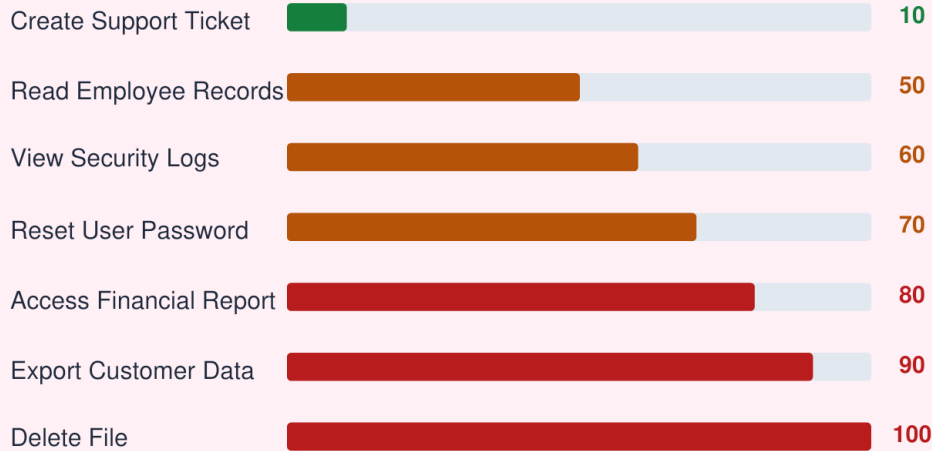
This demonstrates least privilege. A recognised user does not automatically have access to every action.

For example, the employee can create a support ticket because that action exists in the permission list. But if the employee requests customer-data export, identity verification succeeds while the permission check fails. The request is denied before normal risk evaluation and tool execution.

This separation between identity and authorisation is an important Zero Trust principle.

12. Risk Engine and Policy Decision

Different actions have different security impact



Policy can return

ALLOW

REQUIRE
APPROVAL

DENY

The policy stage is reached only after identity and permission checks succeed.

The Risk Engine gives each supported action a predefined numerical score. The purpose is to represent relative impact inside the prototype.

Creating a support ticket has a low score of 10. Reading employee records is 50. Viewing security logs is 60, password reset is 70, financial report access is 80, customer-data export is 90, and file deletion is 100.

These numbers are not a formal enterprise risk calculation. They are demonstration values so that the policy layer can treat routine and sensitive operations differently.

The Policy Engine is reached only after identity and permission checks succeed. It can return Allow, Require Approval or Deny. This adds another control after basic authorisation.

13. Evaluation Design

Controlled scenarios test identity, permission, risk, policy and final enforcement

Legitimate requests

Admin / employee create support ticket
Authorised employee-record access

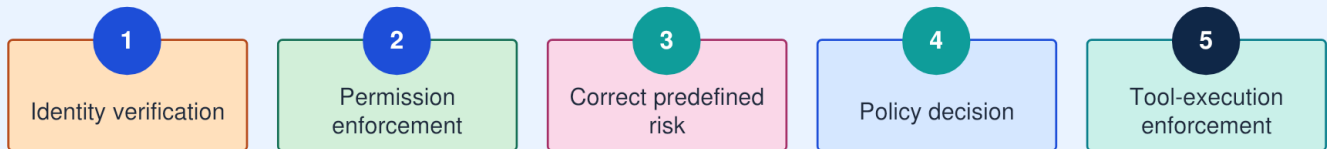
Unauthorised requests

Employee requests customer export, finance report or delete file
Unknown user requests sensitive action

High-risk requests

Financial report, customer export and file deletion
Used to observe policy handling after permission succeeds

Evaluation criteria



The evaluation was performed in a controlled local environment. No real company systems or organisational data were used. The enterprise operations were simulated.

I divided the scenarios into three groups. First, legitimate requests, such as creating a support ticket. Second, unauthorised requests, such as an employee attempting customer-data export or an unknown user requesting a sensitive action. Third, high-risk operations such as finance access, data export and file deletion.

For each case, I considered five things: whether identity was handled correctly, whether permissions were enforced, whether the predefined risk was correct, whether the policy produced the expected decision, and whether the final tool action was actually controlled.

That last point is important because a security decision is only useful if the enforcement point applies it.

14. Main Findings from the Prototype

The evaluation shows clear differences between legitimate access, permission failure and unknown identity

Legitimate request

Employee + Create Support Ticket

- recognised user
- permission exists
- low action risk
- can continue under an allowing policy

Permission violation

Employee + Export Customer Data

- recognised user
- permission does not exist
- Deny - Permission Failed
- export should not continue

Unknown user

Unknown username + sensitive action

- identity verification fails
- Deny - Unknown User
- request stops at identity stage

What this demonstrates

- Identity and permission are separate controls.
- Risk scores distinguish routine actions from high-impact actions.
- The framework separates request generation from execution authority.

The prototype demonstrates several clear behaviours. A legitimate support-ticket request from an employee can pass the identity and permission stages because the action belongs to that role.

A different result appears when the same employee requests customer-data export. The employee is a recognised user, but the permission check fails, so the decision engine returns Deny - Permission Failed.

An unknown username is handled differently. The request stops at the identity stage and returns Deny - Unknown User.

These scenarios show that identity and authorisation are separate. The risk values also give the framework a way to distinguish routine and high-impact actions.

I deliberately did not report numerical success percentages because the complete observed test set was not available. Reporting assumed values would have been misleading.

15. Contribution and Limitations

What the study adds — and what it does not claim

Main contribution

- A security boundary between AI-agent decisions and tool execution.
- One workflow combining identity, RBAC, risk, policy, logging and enforcement.
- A working Python prototype that demonstrates the design in code.
- Controlled evaluation across different users, permissions and actions.

Main limitations

- Keyword-based agent, not a production LLM planner.
- Static action risk scores, not contextual real-time risk.
- Predefined users; no enterprise identity provider or MFA.
- Simulated tools and a small controlled test set.
- Does not directly detect every prompt-injection attack.

The main contribution is the security boundary itself. The framework treats the AI Agent output as a request, not as automatic authority to act.

It also combines several controls in one workflow: identity verification, RBAC, action risk, policy evaluation, logging and final enforcement. The Python prototype makes the idea practical rather than purely conceptual.

At the same time, the limitations are important. The AI Agent is keyword-based. The risk model is static. The users and tools are simulated. There is no enterprise identity provider, MFA or large-scale evaluation.

The framework also does not directly detect every prompt-injection attack. It mainly controls the resulting action at the point of execution. So it should be viewed as one layer in a wider defence-in-depth architecture.

16. Conclusion and Future Work

Three takeaways to remember

1

AI capability $\neq$ authority

An agent may be able to request an action without being allowed to execute it. The request is treated as a security request, not as automatic permission.

2

Zero Trust can protect the tool boundary

Identity, permission, risk and policy can be checked before execution.

3

Independent enforcement matters

The security decision must be applied before the enterprise tool runs.

Future work

- Integrate a real LLM-based autonomous agent and realistic enterprise APIs.
- Use contextual and continuous risk assessment.
- Connect enterprise identity, MFA and a real human approval workflow.
- Run larger adversarial tests, including prompt injection and multi-agent scenarios.

To conclude, I want to leave three main messages.

First, the ability of an AI agent to request an action should not be treated as authority to execute that action. Second, Zero Trust principles can be adapted to the AI-agent tool boundary by checking identity, permissions, risk and policy before execution. Third, the final enforcement point is essential because a decision only protects the system if it is actually applied before the tool runs.

The next step would be to integrate a real LLM-based agent, use contextual risk, connect to enterprise identity and MFA, implement a real human approval workflow, and run larger adversarial evaluations.

Within the limits of the prototype, the study demonstrates a practical way to separate AI-generated intent from execution authority. Thank you, and I am happy to answer questions.

17. Literature References — 1 of 2

Sources used in Chapter 2 of the dissertation

- [1] A. Chhabra, S. Datta, S. K. Nahin, and P. Mohapatra (2026), “Agentic AI Security: Threats, Defenses, Evaluation, and Open Challenges,” IEEE Access.
- [2] National Institute of Standards and Technology (2025), “Strengthening AI Agent Hijacking Evaluations.”
- [3] W. Ma et al. (2026), “OpenClaw in the Wild: Security Analysis of Autonomous Agents,” IEEE/CAA Journal of Automatica Sinica.
- [4] S. Rose, O. Borchert, S. Mitchell, and S. Connelly (2020), Zero Trust Architecture, NIST Special Publication 800-207.
- [5] S. Nashwan et al. (2025), “Autonomous Agents in a Zero-Trust World: Architecting Secure, Auditable AI Workflows,” ITT 2025.

18. Literature References — 2 of 2

Sources used in Chapter 2 of the dissertation

- [6] D. B. Acharya, K. Kuppan, and B. Divya (2025), “Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey,” IEEE Access.
- [7] Y. He, E. Wang, Y. Rong, Z. Cheng, and H. Chen (2025), “Security of AI Agents,” IEEE/ACM Workshop on Responsible AI Engineering.
- [8] S. K. Venugopal et al. (2025), “Enterprise AI Agents: Secure, Scalable, and Autonomous Intelligence for the Modern Workforce,” ICCED 2025.
- [9] A. Doshi et al. (2026), “Towards Verifiably Safe Tool Use for LLM Agents,” ICSE-NIER 2026.
- [10] X. Jiang et al. (2026), “Agentic AI as a Cybersecurity Attack Surface: Threats, Exploits, and Defenses in Runtime Supply Chains,” IEEE CAI 2026.