Degree Project in System Engineering

Second cycle, 30 credits

Identity and Access Management for Autonomous Agentic Systems

A Control-Theoretic Approach with Learned Behavioral Monitoring

ALI GHASEMI

Identity and Access Management for Autonomous Agentic Systems

A Control-Theoretic Approach with Learned Behavioral Monitoring

ALI GHASEMI

Master's Programme, System Engineering, 120 credits
Date: August 24, 2026

Supervisors: Paris Carbone, Nicolae Paladi
Examiner: Amir H. Payberah
School of Mathematics
Host company: CanaryBit

Abstract

Autonomous AI agents challenge classical Identity and Access Management (IAM) because valid credentials do not guarantee that an agent's behaviour remains aligned with the intent under which access was granted. Stochastic reasoning, persistent memory, and autonomous tool use may cause an agent to drift during a session without any identifiable credential compromise. This work therefore reformulates runtime access control as a behaviour-driven control problem in which privileges are adapted according to observed agent trajectories. A graph-structured enterprise resource environment is developed to generate benign and adversarial interactions. The general framework considers behavioural risk, historical trust, contextual information, action criticality, and autonomy state. Its experimental instantiation uses risk and historical trust to select one of four graded autonomy roles. The controllers are evaluated through offline replay of 444 benign and 145 adversarial episodes, comparing a linear soft-threshold policy, a discrete finite-horizon quadratic-cost controller, and a barrier-inspired safety filter. On the primary detection regime, consisting of successful and partial attacks not fully stopped by the static auditor, the behavioural model achieves a ROC-AUC of 0.888 and a PR-AUC of 0.937, improving ROC-AUC by 0.243 over a metadata-only baseline. Historical trust particularly improves successful-attack discrimination, raising ROC-AUC from 0.751 to 0.805. In the single-agent, discrete-role setting, the discrete finite-horizon controller provides no measurable advantage over the linear policy, while the safety net captures most of the achievable safety–utility trade-off. Under replay evaluation, the integrated pipeline retrospectively denies approximately 94% of sensitive actions occurring in attack episodes while allowing 75–80% of sensitive benign actions. These results demonstrate the value of trajectory-level monitoring, history-aware trust, and graded privilege adaptation, while identifying stealthy attacks, offline replay, and the absence of cross-agent coupling as the main limitations.

Keywords
Agentic AI, Identity and Access Management, Risk and Trust-Based Access Control, Behavioral Monitoring, Conditional Variational Autoencoder, Model Predictive Control, POMDP

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Motivation

The contemporary enterprise environment is undergoing a structural transformation, shifting from deterministic software execution models to non-deterministic, autonomous multi-agent systems—systems capable of reasoning, planning, and executing multi-step workflows with minimal human oversight—powered by generative artificial intelligence (GenAI). This evolution represents a fundamental change in how digital actions are initiated, processed, and governed, necessitating a corresponding transformation in Identity and Access Management (IAM).
Traditional security paradigms are proving increasingly inadequate. These agents do not merely respond to prompts in a stateless manner; they accumulate context across interactions, exhibit emergent behaviors during execution, and operate across complex, cross-system environments [1]. The emergence of the “Agentic Web” therefore creates an urgent need for specialized IAM frameworks tailored to Autonomous Non-Human Identities (A-NHIs).
Historically, IAM functioned as a gatekeeper, verifying human identities and a limited set of static machine workloads to grant access to well-defined resources. As AI agents evolve into digital employees capable of self-directed goal formation, persistent memory accumulation, and dynamic tool orchestration, the foundational assumptions of authentication and authorization are placed under strain. Agentic systems introduce novel security vulnerabilities, most notably persistent prompt injection and memory poisoning, which exploit stateful reasoning to achieve compromise without traditional malware [2].
Existing standards bodies such as NIST, ISO, and OASIS model IAM from complementary perspectives. Synthesizing these approaches allows IAM to be structured into four core layers: (1) identity verification, (2) credential and token management, (3) authorization decision logic, and (4) governance and lifecycle control. However, these layers implicitly assume deterministic subjects. In the first two layers, protocols such as OAuth 2.1 and OpenID Connect were designed for deterministic clients operating under explicit user delegation. This assumption collapses in the presence of stochastic reasoning and autonomous planning, where a single malicious instruction may hijack execution across sessions. At the authorization layer, models such as Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) lack the granularity and adaptivity required to govern entities whose behavior evolves dynamically.

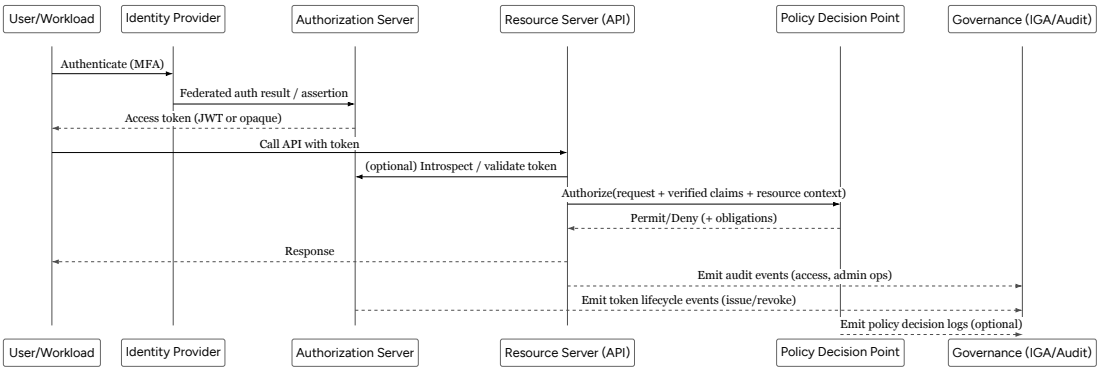

Figure 1.1.1: Conventional IAM architecture showing authentication, token issuance, authorization, resource access, and governance logging for deterministic users and workloads.

Consequently, a new paradigm is required. Autonomous AI agents introduce non-deterministic, evolving behavior that demands an additional runtime governance layer capable of continuous behavioral monitoring and dynamic privilege adaptation. To address this attribution gap, this work proposes a control-theoretic IAM architecture that combines stochastic behavioral modeling with optimal control. In this framework, agent behavior is monitored in real time, and access privileges across the foundational IAM layers are adjusted dynamically in proportion to observed risk.

1.2 Problem Statement

Classical Identity and Access Management (IAM) was designed around two foundational assumptions: that the subject requesting access is a deterministic entity whose behavior can be characterized by static credentials, and that authorization decisions can be made at a single point in time based on identity, role, or attribute. Both assumptions break down in the presence of autonomous AI agents.

Agentic systems are stochastic at multiple layers. At the model layer, generation is probabilistic; at the system layer, reasoning trajectories and tool selection vary across executions; and at the environment layer, interaction with external resources introduces additional non-determinism. As a result, an agent authenticated with valid credentials may, within the same session, exhibit behavior that diverges significantly from the user's original intent—whether due to prompt injection, memory poisoning, emergent reasoning errors, or cascading failures across collaborating agents. This is the *attribution gap*: the actions an agent performs are technically authorized, yet no longer reflect the intent under which authorization was granted.

Existing approaches provide only partial responses to this problem. Dynamic and context-aware access control models introduce adaptivity but treat risk as a point-in-time quantity. Risk-based access control incorporates contextual signals but depends on static weighting schemes and manually engineered risk factors. Runtime governance frameworks such as MI9 and ATF advance continuous monitoring but remain predominantly reactive, responding to observed deviations rather than anticipating them. None of these approaches provide a principled, predictive mechanism for regulating the *trajectory* of agent behavior over time.

The core problem addressed by this thesis is therefore the absence of an IAM framework capable of (i) representing agent behavior as a stochastic, temporally-evolving process; (ii) estimating risk and trust continuously from

observed behavior rather than static attributes; and (iii) adjusting access privileges proactively based on predicted future behavior, subject to safety and policy constraints. Addressing this problem requires reformulating access control itself—from a discrete authorization decision into a closed-loop control process.

1.3 Research Questions

This thesis is guided by the following main research question:

> **RQ (Main).** How can Identity and Access Management for autonomous AI agents be reformulated as a closed-loop control problem, such that access privileges are continuously adapted to an agent's observed behavior rather than fixed at authentication time?

This main question is decomposed into four sub-questions, each corresponding to a core component of the proposed framework:

RQ1 (Behavioral Representation). How can the runtime behavior of an autonomous agent be modeled as a stochastic, temporally-evolving process suitable for online risk estimation, and to what extent does a CVAE+LSTM architecture capture deviations from learned normal behavior in a multi-agent environment?

RQ2 (Trust as Filtered History). How can historical trust in an agent be formalized as a belief over latent safety states using a Partially Observable Markov Decision Process (POMDP), and does this formulation detect slow, subtle behavioral degradation that point-in-time risk scores miss?

RQ3 (Adaptive Access-Control Decisions). How can instantaneous behavioural risk and history-aware trust be translated into adaptive autonomy roles within a control-theoretic access-control framework, and how do different role-selection policies trade off attack containment against preservation of benign utility under offline replay?

RQ4 (Integrated Experimental Evaluation). When behavioural risk estimation, belief-based trust filtering, and adaptive role assignment are integrated into a single experimental pipeline, what safety–utility trade-offs are observed, and what limitations remain before the proposed framework can be evaluated as a true closed-loop system?

1.4 Purpose and Significance

The purpose of this thesis is to develop and evaluate a behavior-driven, control-theoretic framework for Identity and Access Management in autonomous AI agent systems. The framework integrates stochastic behavioral modeling, probabilistic trust estimation, and optimal control to enable continuous, predictive regulation of agent privileges at runtime.

The significance of this work is threefold. First, it addresses a concrete and growing gap in enterprise security: as organizations deploy autonomous agents with access to sensitive systems, existing IAM infrastructure lacks the mechanisms to constrain agents whose behavior may deviate from intent without any identifiable credential

compromise. Second, it contributes a methodological bridge between two fields that have rarely been combined in the access control literature—deep probabilistic behavioral modeling and control theory—by showing that access decisions can be formulated and solved as a constrained optimal control problem. Third, it provides an empirical foundation for evaluating behavior-aware IAM, including a reproducible simulation environment, a dataset of benign and adversarial agent trajectories, and a set of baselines and metrics against which future work can be compared.

1.5 Goals and Objectives

The overall goal of this thesis is to propose a closed-loop, behaviour-aware IAM architecture and evaluate a simplified experimental instantiation through offline replay. This goal is decomposed into the following concrete objectives:

1. Develop a threat model for autonomous AI agents that identifies the protected assets, adversary profiles, attack surfaces, and trust boundaries relevant to agentic IAM.
2. Design and implement a simulation environment representing a structured enterprise resource graph, together with a multi-agent workflow capable of generating both benign and adversarial behavioral trajectories.
3. Construct a behavioral model based on a Conditional Variational Autoencoder with an LSTM backbone, capable of learning temporal patterns of normal agent behavior and producing a continuous instantaneous risk signal.
4. Formalize historical trust as a belief over latent safety states using a POMDP, and integrate it with the instantaneous risk signal to form a unified behavioral state representation.
5. Implement a simplified experimental instantiation of the proposed framework using reconstruction-based risk estimation, belief-based historical trust, and adaptive autonomy-role assignment.
6. Evaluate the integrated pipeline retrospectively through offline replay and quantify its safety–utility trade-off on benign and attack episodes.
7. Identify the methodological and experimental limitations that must be resolved before evaluating the full framework in a genuinely closed-loop environment.

1.6 Delimitation

This thesis focuses on behavior-aware, runtime-adaptive access control for autonomous AI agents and deliberately narrows its scope along several dimensions. The work targets neural, generative agentic systems—in particular, LLM-based agents operating with tool use and inter-agent communication—rather than symbolic or purely reinforcement-learning agents. The threat model considers three representative attack classes drawn from the OWASP Top 10 for Agentic Applications—goal hijacking, memory poisoning, and cascading failures—chosen because they most directly exercise the assumptions that classical IAM fails to enforce. Other attack vectors are acknowledged in the threat model but are not the focus of empirical evaluation.

On the infrastructure side, this thesis does not implement hardware- backed security primitives such as Trusted Execution Environments, nor does it propose or evaluate new cryptographic identity protocols. Although such mechanisms are relevant to a complete agentic IAM architecture, they are treated as complementary and out of scope for the behavioral and control-theoretic contribution presented here.
Finally, the experimental evaluation is conducted in a simulated enterprise resource graph rather than a production deployment. This choice is motivated by the need for controlled, reproducible scenarios and the absence of public datasets containing agent behavior traces paired with ground-truth attack labels. The implications of this design choice—including the generalizability of results and the transfer to real deployments—are discussed in Chapter 8.

1.7 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces agentic AI as a system-level paradigm, examines its architectural characteristics, and analyzes the security implications of non-determinism, emergence, and long-horizon execution. Chapter 3 reviews classical and modern Identity and Access Management, including static, dynamic, context-aware, and risk-based access control models, and motivates the shift toward a behavior-driven, control-theoretic perspective.
Chapter 4 develops the threat model for autonomous agents, synthesizing established taxonomies with an architectural decomposition based on the MAESTRO and ATFAA frameworks. Chapter 5 formalizes the proposed closed-loop IAM framework, including behavioral modeling, trust estimation, and MPC-based access control.
Chapter 6 describes the experimental environment, scenario design, agent architecture, and data generation pipeline. Chapter 7 presents the evaluation methodology and results. Finally, Chapter 8 summarizes the contributions and outlines directions for future work.

2 Agentic AI

The emergence of agentic Artificial Intelligence (AI) represents a shift from static, inference-based systems toward autonomous, goal-directed computational entities. Unlike traditional AI models that perform isolated predictions, agentic systems are persistent actors that iteratively perceive, reason, and act within dynamic environments. This redefines the role of AI from a passive analytical tool into an active participant in complex digital ecosystems.

This chapter establishes the kind of system the rest of the thesis is concerned with. It first defines agentic AI and its architectural components, then characterises the structural properties—non-determinism, emergence, and temporal evolution—that distinguish agentic systems from both static models and traditional software workloads. It closes by sketching the security implications that motivate the more detailed treatment in Chapter 3 and Chapter 4, and by arguing that the trajectory of the field toward fuller autonomy and physical embodiment makes a general, behaviour-aware security approach necessary.

2.1 What is Agentic AI

Agentic AI is positioned in the literature as a shift away from passive, task-specific AI tools toward autonomous systems that exhibit *agency*—the capacity to pursue goals through iterative perception, reasoning, and action in an environment, rather than producing a single static prediction or response [3, 4]. Modern agentic systems are characterised by proactive planning, contextual and often persistent memory, sophisticated tool use, and adaptation based on environmental feedback [3].

A useful conceptual boundary, particularly for security analysis, separates agentic AI systems from AI *models*. A foundation model—typically a Large Language Model (LLM)—may be central to an agent, but agentic behaviour depends on system-level modules (planning, memory, retrieval, tool execution) that transform a model into an actor operating over time [4, 5]. Intelligence is therefore not only inferential (mapping input to output) but also *operational*: selecting and executing actions under uncertainty [3, 4].

A second useful boundary is between an individual agent and *agentic AI* as an architectural approach that orchestrates multiple specialised agents into coordinated workflows [3]. This orchestration framing is important because it highlights that agentic AI is not merely a larger model; it is a larger *system* whose behaviour emerges from interactions among components such as planner, memory, tools, and inter-agent communication [3, 4].

The integration of autonomous AI agents into enterprise infrastructure represents a structural shift in digital interaction [6]. Organisations are moving beyond experimental use of LLMs toward the deployment of agents capable of executing multi-step workflows with minimal human oversight [7]. Current projections indicate that 25% of enterprises are expected to adopt autonomous AI pilots by 2025, increasing to 50% by 2027 [6]. As these agents interact with multiple systems and adapt to environmental feedback, their behaviour becomes increasingly difficult to

predict and constrain using traditional system assumptions [1, 8].
At the architectural level, agentic systems implement an iterative control loop—observe, plan, act, update—rather than a single inference step [4, 9]. Across surveys, the recurring core modules are (i) a planning and decomposition layer that converts high-level objectives into action sequences, (ii) memory and retrieval mechanisms for temporal continuity and grounding, and (iii) tool-use mechanisms that allow interaction with external resources such as APIs, software, web content, or embodied actuators [4, 5, 3]. The practical significance is that agentic AI can pursue complex goals in dynamic environments by chaining actions and reacting to feedback, which is one reason these systems are increasingly considered for high-stakes settings [3, 9].
A related architectural trend is the rise of *world models* as internal representations supporting planning and decision-making, by encoding either the present state or predicted future dynamics [10]. Because agents frequently operate in environments where exploration is costly or risky, world models enable agents to plan with foresight rather than only reacting [10]. As these models become more capable—driven by progress in multimodal foundation models—the action space an agent can reason over broadens, expanding both autonomy and the scope of possible failure modes [10].

2.2 Non-determinism, Emergence, and Temporal Evolution

Three structural properties of agentic systems matter for the security argument that follows: non-determinism, emergent behaviour, and temporal evolution. They are introduced together here because they are coupled: each amplifies the others.
Modern agentic systems—especially neural agents—are non-deterministic at multiple layers [3, 4]. At the model layer, stochastic generation is a defining characteristic of the neural lineage of agentic AI [3]. At the system layer, agents exhibit stochastic reasoning and variable tool execution outcomes [4]. At the environment layer, tool execution variability and environment non-determinism create challenges for reproducibility, comparability, and interpretability [4]. The practical consequence is that two executions of the same agent against the same prompt can produce materially different action sequences.
This multi-layer non-determinism is closely coupled with emergent behaviour, especially in multi-agent settings. Even small groups of interacting agents can generate a vast number of possible interaction patterns, leading to emergent behaviours that are not explicitly programmed by system designers [11].
Evaluation-oriented surveys trace how multi-agent systems force evaluation to include cooperation, competition, and emergent behaviour, because the system outcome is not reducible to any single agent's local policy [4]. Emergent coordination patterns can therefore produce unanticipated action sequences, making it difficult to guarantee that a system will remain within intended operational boundaries using static rules alone [4, 12].
Unpredictability is further reinforced by temporal evolution: agentic systems do not remain static after deployment. A systematic literature review identifies a scarcity of architectural frameworks supporting long-term resilience and dynamic behaviour modification in unforeseen environments [12]. The self-evolving agents survey emphasises that many deployed agents are effectively static after deployment, motivating new approaches that automatically enhance agent behaviour using

interaction data and environmental feedback [5]. Unpredictability is therefore not merely an incidental artefact of stochastic inference; it can be an intended property of systems designed to improve continuously post-deployment [5, 12].
Together, these properties mean that a security guarantee made at deployment time does not transfer to behaviour at runtime: the system that gets certified is not the system that gets executed.

2.3 Trajectory Toward Full Agency and Physical AI

The trajectory of the field is toward greater autonomy and broader operational scope, not lesser. Two trends in the recent literature make this concrete.
The first is the move toward continuously evolving systems. The self-evolving agents survey frames evolution techniques as a way to update prompts, memory, tools, workflows, and inter-agent communication based on feedback loops, presenting self-evolving agents as a bridging paradigm between static foundation models and lifelong agentic systems [5]. A systematic review of the field similarly characterises the present period as rapidly evolving, with an emphasis on operationalising autonomy and a noted gap in adaptability and long-term resilience [12]. The implication is that security controls must remain effective under continual modification, which one-shot certification cannot deliver [4, 13].
The second is the extension of agentic AI into physical and embodied domains. Agentic systems are increasingly deployed in robotics, drones, and autonomous vehicles, where they must perceive, decide, and act under real-world operational constraints [14]. Embodied AI for security highlights challenges that include hardware and computational limits, real-world robustness, ethics and privacy, interoperability with existing systems, and cybersecurity [14]. From a security-design perspective, embodied settings are not a separate category from cyber agents; they share the same core problem—an autonomous actor whose trajectory must be governed at runtime—but with higher consequences when the trajectory goes wrong [14, 3].
Both trends point toward the same design conclusion. A security approach tied narrowly to a specific architecture, deployment context, or threat catalogue will struggle to transfer as agents diversify and as their action spaces expand into the physical world. The work in this thesis is therefore framed around the most general property these systems share: they act as *actors operating on behalf of human intentions*, and the fundamental security problem is keeping their executed behaviour aligned with that delegated intent over time.

2.4 Security Implications

The properties above have direct consequences for how agentic systems can be secured. A major synthesis on foundation agents organises the primary classes of risks and vulnerabilities in agentic systems—prompt injection, hallucination, misalignment, poisoning, and privacy concerns—and frames building safe and beneficial AI agents as a core imperative [13]. This is significant because autonomy and tool use transform classical model-level risks (incorrect or imprecise outputs) into system-level operational risks, where errors can directly lead to unintended actions in real-world or enterprise environments [3, 13]. Prompt injection is one concrete example of this shift: it can hijack behaviour through both direct user

prompts and indirect external content such as web pages or retrieved documents, tying security risk directly to integration patterns rather than to the model alone [13, 4].

The combined effect of stochastic reasoning, emergent multi-agent behaviour, and continuous post-deployment evolution is that traditional, point-in-time security mechanisms cannot fully constrain agentic systems. Static performance metrics and one-shot certification are insufficient; assurance must be statistical, behavioural, and ongoing rather than purely deterministic and pre-deployment [4, 13]. A general security strategy must control *interfaces*—inputs, tool channels, memory updates—rather than only the core model [13, 4]. And because the field spans architectures from symbolic to neural, from cyber to embodied, security solutions need to be architecture-agnostic at the systems level rather than tied to a specific implementation lineage [3, 14].

In the framing of this thesis, securing agentic AI is therefore less like model hardening and more like securing an evolving autonomous software system, where generality and scale come from controlling interfaces, monitoring behaviours over time, and supporting architectural diversity [3, 13, 5, 4, 14]. The detailed treatment of specific threats, attack surfaces, and trust boundaries is deferred to Chapter 4; this chapter has only argued that such a treatment is needed.

2.5 Summary

Agentic AI systems differ from traditional software entities in three structural ways: they are non-deterministic at multiple layers, they exhibit emergent behaviour in multi-agent settings, and they evolve through interaction over time. The trajectory of the field is toward greater autonomy and into physical domains, which makes a narrow, architecture-specific security solution unlikely to transfer. These systems are not passive subjects requesting access to resources; they are active decision-makers that act on behalf of human intentions, and their identity must be understood in terms of behaviour, intent, and execution context rather than static credentials.

This shift challenges the foundational assumptions of classical Identity and Access Management, which were designed for deterministic users and machine workloads with predictable behaviour. The next chapter examines whether existing IAM frameworks can govern this new class of subject, and motivates the shift toward behaviour-driven, control-theoretic access control that this thesis develops.

3 Identity and Access Management

The maturation of classical Identity and Access Management (IAM) has been driven by four converging forces: increasing technological complexity, rapid identity scale expansion, evolving threat sophistication, and growing regulatory pressure. As digital systems progressed from local enterprise networks to cloud-native and distributed architectures, the number and diversity of identities expanded dramatically. Simultaneously, attacks evolved from simple credential theft to complex lateral movement and API abuse, while regulatory frameworks demanded stronger governance and auditability. Together, these forces shaped the foundational structure of modern IAM.

This chapter argues that the existing IAM toolset cannot govern the kind of system Chapter 2 described. The argument has three steps. First, the identity object IAM is built around has changed from human to machine to agent, and existing identity protocols implicitly assume properties agents lack. Second, the access-control models built on those protocols—from role-based to attribute-based, context-aware, and risk-based—have made progress in adaptivity but still treat risk as a point-in-time quantity rather than a property of behaviour over time. Third, the alternative this thesis proposes is a behavioural, control-theoretic formulation, motivated by analogous problems in other domains where decisions must be made under uncertainty over time.

3.1 Classical Identity Management: From IAM to MIM

3.1.1 Human-Centric IAM

In the early stages of digital systems, the primary objective was the verification of human personnel interacting with government or enterprise information systems over local networks [15]. Access was typically granted based on a one-time validation of credentials such as a username and password, assuming that once an entity was authenticated it could be trusted throughout the session [16]. As organisational complexity increased, static permissions assigned to individuals became unmanageable, leading to the development of structured access control models. Role-Based Access Control (RBAC), formalised and promoted by the National Institute of Standards and Technology (NIST) in the 1990s, simplified administration by bundling permissions into roles aligned with job functions [17]. This effectively operationalised the principle of least privilege for human workforces but faced limitations as software became more dynamic.

3.1.2 Machine Identity Management (MIM)

The rise of cloud-native architectures and microservices shifted the focus from human identities to non-human identities (NHIs), which represent applications, service accounts, IoT devices, and automated scripts that operate without direct human intervention [18]. NHIs now outnumber human identities by a factor of 50 to 1, yet they are frequently managed with significantly less rigour than human accounts

[19]. Unlike human subjects, machine identities rely on cryptographic credentials such as Certificate Authority–issued certificates, API keys, and mutual TLS tokens [19]. The management of these identities, known as Machine Identity Management (MIM), focuses on the lifecycle of discovery, classification, issuance, and rotation of credentials [19]. A recurring research gap concerns shadow and orphaned credentials—machine identities created outside formal IAM governance, or accounts that retain high-level permissions long after their primary workload has been decommissioned [19]. The industry response to these challenges is reflected in NIST SP 800-63 Revision 4, which introduces a risk-based Digital Identity Risk Management framework that moves away from checklist-based compliance toward continuous evaluation, defining Identity, Authenticator, and Federation Assurance Levels and emphasising phishing-resistant MFA and verifiable credentials as a Zero Trust prerequisite [20, 21].

3.1.3 Agent-Aware Identity Paradigm

Securing the agentic web requires recognising autonomous agents as a distinct identity class requiring full lifecycle governance and dynamic authentication [8]. Traditional IAM protocols such as OAuth 2.1 and OpenID Connect were designed for deterministic clients operating under explicit user delegation [22]. This assumption collapses in agentic environments, where multi-agent systems share credentials, invoke downstream services autonomously, and alter execution paths dynamically [22].

A central challenge is the attribution gap. In many deployments, agents operate indistinguishably from human users, making it difficult to trace which entity initiated a specific action and under what reasoning context [2]. Standard OAuth tokens cannot express lineage—identifying which agent invoked another and why—limiting forensic traceability and accountability [22]. To address this limitation, advanced identity protocols such as Agentic JWT (A-JWT) introduce cryptographic binding between agent identity, user intent, and workflow state, assigning each agent a unique checksum-derived identity and supporting chained delegation and proof-of-possession mechanisms to prevent unauthorised lateral movement and replay attacks [22]. These developments reflect a broader shift toward agent-aware IAM, where identity must encode intent, lineage, and execution context rather than merely authenticate a static principal.

3.2 Access Control Models

The main focus of this study is the conceptual foundations of identity and access control, rather than governance frameworks or protocol specifications. NIST provides the most formalised and widely cited definitions of RBAC and ABAC and is therefore used as the primary reference for authorisation logic.

A clear distinction is made between authentication, authorisation, and access control. Authentication verifies the identity of a subject; authorisation determines whether an authenticated subject may perform a particular action on a particular resource; and access control is the mechanism that enforces those authorisation policies in practice [23]. An access control model can be described in terms of five elements—subjects, actions, objects, privileges, and policies—and approaches are typically categorised by decision-making behaviour as either static or dynamic [23].

3.2.1 Static Access Control

Static access control models—DAC, MAC, RBAC, ABAC, and PBAC—use predetermined policies that yield the same decision in different circumstances. RBAC bundles permissions into job-function roles [17]; its limitations, particularly the role-explosion phenomenon, motivated the move toward Attribute-Based Access Control (ABAC) in the 2010s, which evaluates attributes such as user department, resource sensitivity, and location at the time of access [24]. Policy-Based Access Control (PBAC) generalises this further by evaluating requests against organisation-defined policies through a policy decision component [25]. Mandatory and Discretionary Access Control (MAC and DAC) round out the classical family: MAC categorises objects by sensitivity level and enforces access through clearance labels, while DAC delegates access decisions to object owners [23].
These models have served their original deployment contexts well, but they share a structural limitation: their static and rigid policies cannot handle unpredicted situations and cannot adapt to dynamic, distributed environments such as IoT and cloud computing, which require contextual flexibility [23]. To address this limitation, dynamic access-control methods incorporate not only static policies but also dynamic and real-time features—context, trust, and security risk—into access decisions [23].

Table 3.2.1: Traditional Access Control Models

Model	Description
DAC (Discretionary Access Control)	Access is based on the identity and authorization of the subject, where the owner of an object has the discretion to grant or revoke access permissions to other users.
MAC (Mandatory Access Control)	Access is determined by security labels assigned to subjects and objects. Objects are categorized according to sensitivity levels, and subjects are granted access based on their clearance level.
RBAC (Role-Based Access Control)	Access is based on user roles within an organization. Roles are assigned permissions according to job functions and responsibilities.
ABAC (Attribute-Based Access Control)	Access is determined by attributes related to the user, resource, and environment, which can be evaluated at the time of access.
PBAC (Policy-Based Access Control)	Access decisions are determined by evaluating a set of security policies defined by the organization. These policies specify the conditions under which a subject is allowed to access a resource.

3.2.2 Dynamic, Context-Aware, and Risk-Based Access Control

Dynamic access-control models consider not only access policies but also dynamic and contextual features collected at the time of the access request. This provides flexibility and allows decisions to adjust to varying circumstances [23]. The improvement over static models is significant, but dynamic models introduce their own limitations: they are more complex to design and implement, the relevant contextual attributes vary across domains, weight assignment is subjective, and processing dynamic context introduces computational overhead [23].

Context-Aware Access Control (CAAC) is one of the principal approaches for implementing dynamic access control. The notion of context awareness originates in ubiquitous computing, where context is defined as any information that can be used to characterise the situation of a person, place, or object relevant to an interaction [25]. CAAC extends RBAC and ABAC by enabling access decisions to adapt to changing contextual conditions through a four-stage lifecycle—context acquisition, modelling, reasoning, and dissemination—and has been applied across cloud, enterprise, healthcare, and IoT domains [25]. From an access-control perspective, context can be classified into user-centric, resource-centric, and environment-centric categories [25]. Higher-level access-control contexts can be inferred from lower-level contexts, which is particularly relevant for handling uncertainty and probabilistic scenarios when estimating risk [25]. This work adopts these well-established context-classification and lifecycle concepts as a starting point for the agentic-AI setting.

Risk-based access control extends conventional access control by evaluating the risk associated with each request rather than relying solely on fixed rules. Risk is defined as a measure of the extent to which an entity is threatened by a potential event, based on the likelihood and impact of that event, and the request is permitted only when the estimated risk remains below an acceptable threshold [23]. Risk-based models can incorporate contextual, environmental, situational, mission-related, and trust-related factors, and may include risk-mitigation mechanisms such as additional obligations or controls applied before or after access is granted [23, 25].

Trust enters this picture as a complementary factor. Trust represents the level of confidence a resource controller has that a subject will not misuse the accessed resource [25]. In trust-based CAAC, evidence may be derived from behavioural history, reputation, or security indicators; multidimensional trust models compute trust using mechanisms such as fuzzy logic and classify entities into multiple trust levels [25]. Quantified and adaptive variants such as QRACC map decisions onto multiple risk bands, each associated with a different mitigation action such as increased auditing or sandboxing, and architect the system around explicit Trust Estimation, Risk Estimation, Adjustment, and Risk-Aware Access Control modules [25]. The argument that emerges is that access decisions should result from a balanced assessment of both risk and trustworthiness rather than from static authorisation rules alone [25].

This thesis adopts that argument as its starting point. Static models such as RBAC and ABAC are not adequate for agentic systems, and the appropriate direction is risk-aware and trust-aware access control. The remainder of this chapter examines what behavioural grounding is needed to make risk and trust meaningful in this setting, and how analogous problems have been solved in other domains.

3.2.3 Risk Estimation in Risk-Based Access Control

Mathematically, the most common formulation expresses risk as the likelihood of an incident occurring multiplied by its impact [23]. The risk-estimation stage is the central component of any risk-based access-control model: it analyses access requests, gathers the relevant risk factors, and estimates the security risk value that is then compared against access policies to determine the access decision [23]. Determining the appropriate technique for this stage is non-trivial because the choice depends heavily on the deployment context and on the availability of data describing risk likelihood and impact [23]. A systematic literature review of 44 risk-based access-control studies reports that mathematical-equation approaches are often dependent on domain-specific variables and difficult to generalise; fuzzy-logic systems suffer from subjectivity and require domain experts to define variables and rules; risk-assessment techniques identify and prioritise risks but do not directly produce numeric values; and machine-learning and game-theoretic approaches remain under-explored, mainly due to the lack of suitable datasets [23]. The same review concludes that the availability of datasets describing risk likelihood and impact is one of the key challenges in implementing effective risk-based access-control models, encouraging researchers to build and share such datasets to introduce learning capability into existing approaches [23].

3.3 Zero Trust and Runtime Governance

Traditional perimeter-based security architectures rely on a trusted internal network protected by mechanisms such as Demilitarised Zones and firewalls, but this model becomes vulnerable once an attacker gains internal access [25]. The increasing complexity of modern enterprise environments—driven by cloud computing, BYOD, IoT, edge computing, and remote work—has weakened perimeter-based security and led to more distributed and dynamic architectures [25]. Zero Trust (ZT) emerged as a response: internal and external networks are treated as equally untrusted, every access is validated on a per-session basis, and policy enforcement is dynamic, taking device, user, behavioural, and environmental attributes into account [25]. NIST defines a Zero Trust architecture around five principles: all data and services are resources, access is granted per session and may be reviewed during the session, decisions are determined by dynamic policy on the observable state of the requester, decisions are evaluated by a trust algorithm that may consider history (*singular versus contextual*), and the enterprise continuously monitors asset and environment state [26]. Architecturally, ZT is built around a Policy Engine that decides, a Policy Administrator that establishes or terminates communication, and a Policy Enforcement Point that enforces the decision [26, 25].

In response to AI-specific risks, governance frameworks tailored to agentic systems have been built on top of these principles. The NIST AI Risk Management Framework emphasises AI systems that are valid, reliable, safe, secure, and resilient [27]. The AWS Agentic AI Security Scoping Matrix classifies deployments by levels of connectivity and autonomy, recommending progressively stronger controls as agents approach full autonomy [28]. The Agentic Trust Framework (ATF) operationalises Zero Trust for AI agents by requiring globally unique agent identities, behavioural monitoring, data governance controls, segmentation, and auditable human override [29].

Practical runtime-governance frameworks complement these governance documents. The MI9 framework shifts assurance from pre-deployment validation to live, in-session oversight through behavioural telemetry, continuous authentication, drift detection, and graduated containment [30]. Programmable privilege control systems such as Progent demonstrate the feasibility of deterministic, fine-grained enforcement at the tool-call level, intercepting runtime tool invocations and applying argument-level constraints to provide symbolic guarantees independent of probabilistic LLM reasoning [31].
These frameworks are closer to what agentic IAM needs than classical access control, but they remain largely reactive and rule-based. None of them provides a formal framework for modelling behavioural evolution and proactively regulating agent actions over time. The continuous-evaluation machinery is in place; what is missing is a principled way to turn observed behaviour into a forward-looking policy signal.

3.4 Limitations of Existing Models

The limitations of risk- and trust-based access control models stem primarily from their complexity and the difficulty of accurately modelling dynamic human and environmental factors [32]. While these models offer more flexibility than static policies, they face significant hurdles in implementation and consistency [33]. The four limitations identified below define the gap that the methodology in Chapter 5 is built to address. They are organised so that each maps to one design choice in the proposed framework: behavioural grounding (the CVAE+LSTM detector), temporal reasoning (the POMDP trust filter), dynamic risk assessment (the closed-loop integration), and quantification (the calibrated risk signal).

3.4.1 Lack of Behavioural Grounding

A major limitation of risk- and trust-based models is that they often rely on credentials that do not strictly bind a user to their actual behaviour [34]. Credentials verify identity at a point in time but convey no information about the bearer's behaviour between issuance and use [34]. Many models are context-insensitive, ignoring the activity patterns and intent needed to detect insider threats or account takeovers [32]. Static policies built by security analysts are unable to resolve unidentified threats in real time because they can only address problems that were recognised before the policy was written [23].
Several studies argue that behaviour is needed to reduce misuse by authorised users but stop short of specifying validated behavioural models. The adaptive IoT model in [35] explicitly frames this gap: once access is granted, traditional approaches have limited ability to prevent misuse by an authorised subject; the proposed mitigation is to observe behaviour and adjust permissions, but the behavioural model itself is not formally defined at the policy level [35]. The trust-score model in [32] formalises user behaviour as a factor within a trust score, but behaviour remains a component score derived from operational sub-metrics rather than an explicit behavioural process model [32].
The consequence of weakly grounded behaviour—credential proxies, coarse history, undefined abnormality—is that the access decision becomes brittle: the policy engine cannot robustly distinguish between (i) benign novelty and malicious novelty, (ii) authorised-but-misusing insiders and authorised legitimate users, or (iii) transient

anomalies and persistent compromise. If the underlying behavioural classifier is weak or poorly grounded, both false positives and false negatives increase. Behaviour monitoring is therefore not only essential for dynamic authorisation; the *characteristics* of the behavioural model materially affect the resulting decisions.

3.4.2 Weak Temporal Reasoning

Many access-control systems perform authentication only at the initial point of access, lacking continuous validation throughout a session [32]. Even in dynamic models, time tends to be treated quasi-statically: as a context attribute at decision time, as a session boundary for repeated checks, or as a coarse history scalar [36]. While Zero Trust principles advocate for *always verify*, many practical implementations fail to reflect rapidly changing threats or behaviour in real time [32]. With the emergence of agents that have long-context memory and attack vectors such as indirect prompt injection, this problem becomes directly relevant in the agentic-AI setting [34].
When the temporal model is limited to per-session rechecks and pointwise thresholds, the policy engine struggles to detect risk that emerges from action ordering, repetition, and timing patterns. The success of any such system depends on both detection precision and detection timing; time is not a background variable but is coupled to detection performance [37]. Systems that do not model timing and sequences explicitly risk policy lag: by the time a per-session check triggers, the threat may have already progressed.

3.4.3 Dynamic Risk Assessment Limitations

Dynamic risk assessment is frequently hindered by computational complexity and resource burdens, especially in constrained environments such as IoT [36]. In collaborative settings such as Bring Your Own Device, models can become unqualified for new shared scenarios because they lack the specific contextual experience needed for accurate decisions [37]. If the underlying data is heavily biased—for instance, if the majority of historical decisions were positive—automated models tend to predict positive outcomes and fail to detect subtle abnormalities [37]. Scalability and computational cost are first-class concerns in data-intensive, highly dynamic environments: AI-, fuzzy-, and neuro-fuzzy–based approaches provide accuracy and adaptability but face resource demands and scalability challenges [38].

3.4.4 Limitations in Quantifying Risks

Quantifying risk is inherently difficult because there is no global agreement on the meaning of trust or trustworthiness [37]. Four quantitative challenges recur across the literature. First, subjectivity: it is often impossible to define precise semantics for the numbers used [37]. Second, weight assignment: there is significant subjectivity in assigning weights to contextual features, and poorly chosen weights produce inaccurate risk scores [23]. Third, data scarcity: the lack of comprehensive datasets describing risk likelihood and impact prevents many models from generating precise numeric risk values [23]. Fourth, incomplete information: when the risk-estimation process is based on imprecise or incomplete information, identifying the true value of the protected information becomes difficult [23].

3.5 Behavioural Grounding from Adjacent Domains

The pattern of behavioural modelling, anomaly and risk scoring, and trust aggregation is not specific to access control. It recurs in domains that share three structural properties: (i) high-volume event streams, (ii) evolving context, and (iii) asymmetric costs of mistakes (false accept versus false reject) [38, 39, 37]. These are the core characteristics of agentic systems, and the same properties are present in financial markets, IoT, and IoV—domains in which the modelling and decision-making patterns this thesis adopts have been independently developed. The cross-domain analogies are therefore used here to motivate the chosen design rather than to claim novelty for the framing itself.

In finance-oriented transaction environments, security events occur asynchronously during transaction execution and carry direct financial loss implications. A formal model for transaction processing under threat emphasises that intrusion detection is central, that false positives and false negatives shape risk outcomes and decision-making, and that risk mitigation can be framed through sequential decision strategies in which observations are partially informative and decisions must balance security and operational utility [37]. Although the model is presented for financial transaction control, it is explicitly motivated as applicable to other domains in which human error, malfunction, or external intervention events introduce uncertainty—precisely the conditions that motivate behavioural trust updates [37].

In IoT systems, the recurring characteristics are openness, heterogeneity, and sensitivity to deployment circumstances. Static access control does not match dynamic requirements, and risk-based access control therefore emphasises real-time contextual inputs and continuous behavioural monitoring as adaptive features [40, 41]. A survey of fourteen IoT risk-based access-control models reports that AI-enhanced methods achieve precise risk estimation and adaptivity but frequently introduce resource demands and scalability concerns, while resource-efficient designs may trade off security effectiveness or adaptability [38]—a deployment-versus-capability trade-off that recurs in agentic settings.

In IoV and connected-vehicle settings, two complementary characteristics emerge. Connected-car ecosystems collect personal data through in-vehicle technologies and companion applications, creating empirically measurable privacy and cybersecurity concerns at population scale [37]. The IoT risk-based access-control survey describes the use of AI-based approaches for risk prediction and real-time assessment in vehicular networks, highlighting that IoV inherits IoT's dynamism but adds mobility and safety-critical interactions [38].

Closer to the technical mechanism this thesis uses, several systems explicitly couple anomaly detection to enforcement. Behavioural analytics is described as detecting anomalies relative to established baselines, with anomalies treated as triggers for security response and access-control optimisation [42]. A dynamic access-control design uses activity-log and context analysis to identify anomalies such as irregular login times or unfamiliar devices, producing risk scores that drive permission adjustments [43]. A log-based formulation in [44] is particularly explicit: rather than requiring binary anomalous-versus-normal labels, the stated requirement is a scoring function that describes how risky a user access could be, positioning anomaly detection outputs as a continuous risk measure [44]. Stacked LSTM layers are trained to predict the next event identifier from previous events, with the learning objective

explicitly tied to capturing user behaviour patterns over time [44]. Continuous-authentication research applies similar deep-learning classifiers—including multilayer perceptrons and convolutional LSTM architectures—to behavioural-biometric signals such as typing dynamics, while noting practical drawbacks such as complexity and resource intensity in constrained environments [45].

A particularly relevant trust formalisation appears in mobile-device communication under Zero Trust [39]. A dynamic trust evaluation model combines static attributes with historical interaction data, defines trust computation using direct and indirect trust, and introduces a time-decay factor alongside a sliding-window strategy for dynamic threshold updates [39]. This is a concrete mechanism for making "trust is history" precise in an implementable way: recent events matter more, and thresholds adapt as behaviour shifts [39]. The cloud authorisation model of [36] similarly introduces a user behaviour risk value and a user trust degree, and explicitly maps trust levels to permissions—turning trust into a control variable that mediates authorisation [36].

A final caveat: risk depends not only on detection but on detection error characteristics. False positives and false negatives materially affect security risks and countermeasure decisions during transaction processing under threat [37]. Any risk-from-anomaly pipeline must therefore consider error trade-offs explicitly, because trust updates and access restrictions are sensitive to these errors over time [37].

The cumulative implication of these adjacent-domain patterns is that the right design for agentic IAM is not a new framework invented from scratch, but a synthesis: behavioural modelling as a sequence model, risk as a continuous calibrated signal derived from the model, trust as a temporally-filtered aggregate of risk evidence, and access decisions as control outputs informed by both. This is the synthesis Chapter 5 formalises.

3.6 Toward a Control-Theoretic IAM

The limitations identified above suggest that IAM for agentic systems cannot be addressed adequately through static policies or purely reactive enforcement. A framework is needed that reasons about behaviour as it evolves over time, under uncertainty, and in interaction with complex environments. This motivates a shift toward a control-theoretic perspective on IAM.

At the conceptual level, agentic systems can be interpreted as dynamic systems whose internal state—goals, memory, intermediate reasoning—evolves in response to inputs and environmental feedback. Access-control decisions are no longer discrete authorisation events, but continuous control inputs that shape the trajectory of agent behaviour. The objective of IAM becomes analogous to regulating a system: keeping the agent within acceptable safety, security, and policy constraints while preserving functional performance.

This perspective enables a natural mapping. Observed agent behaviour acts as the system state; access privileges and constraints act as control inputs that shape future actions; risk and trust metrics act as state-dependent costs guiding the system toward desirable operating regions; and monitoring telemetry provides the feedback that updates state estimates under uncertainty. The advantages of this formulation are its

explicit handling of non-determinism and temporal dependencies—properties intrinsic to agentic AI—and its ability to incorporate constraints such as safety boundaries, policy requirements, and resource limits directly into the decision process. Rather than enforcing constraints after an action is taken, the system optimises control inputs so that future behaviour remains within acceptable bounds, shifting IAM from a reactive enforcement paradigm to a predictive and preventive one.

Building on these principles, a behaviour-aware IAM architecture can integrate continuous monitoring, probabilistic state estimation, and optimal control. Models of agent behaviour learned from interaction data estimate current state and predict future trajectories; control mechanisms then adjust access privileges dynamically, minimising risk while preserving operational utility. This is the framework the next chapter formalises after Chapter 4 establishes the specific threats it must address.

3.7 Research Gap and Motivation

The preceding analysis identifies a coherent gap. Classical access-control models rely on static identities and predefined policies. Dynamic and context-aware approaches introduce adaptability but remain dependent on instantaneous decision logic. Risk-based access control extends the paradigm by incorporating contextual risk signals, but largely treats risk as a point-in-time estimate rather than a temporally evolving property. Recent runtime-governance frameworks emphasise continuous monitoring and enforcement, yet remain predominantly reactive: they respond to observed deviations rather than anticipating them.

Collectively, these approaches lack a unified framework for modelling and regulating the evolution of agent behaviour over time. There is no formal mechanism to (i) represent behavioural trajectories, (ii) predict future states under uncertainty, and (iii) systematically adjust access decisions to guide behaviour toward desired operational constraints. This gap is critical in agentic environments, where autonomy, non-determinism, and long-horizon decision-making introduce dynamics that cannot be managed effectively through static policies or rule-based adaptations alone.

This work addresses the gap by proposing a behaviour-driven, control-theoretic framework for IAM in agentic systems. The central premise is that agent behaviour can be modelled as a stochastic, evolving process, enabling probabilistic models to estimate system state and predict future behaviour. Building on this representation, access control is reformulated as an optimal control problem: permissions and constraints are dynamically adjusted to minimise risk while maintaining functional performance. This approach enables proactive governance by incorporating prediction, feedback, and constraint-aware optimisation into the decision-making process. By integrating behavioural modelling with control-theoretic principles, the proposed framework shifts IAM from a static, reactive paradigm toward a predictive, adaptive, and continuously regulated system. The next chapter formalises the specific threats this framework must address; Chapter 5 specifies the framework itself.

4 Threat Model

As organisations move beyond Large Language Model (LLM) experimentation toward the deployment of agentic systems—entities capable of reasoning, recursive planning, and executing complex workflows with minimal human oversight—the structural assumptions of legacy Identity and Access Management (IAM) have become inadequate. Traditional IAM protocols such as OAuth 2.1 and OpenID Connect were architected on the premise of deterministic clients acting under explicit, well-defined user delegation. This premise collapses when confronted with the stochastic nature of agentic reasoning, where an autonomous entity can independently spawn sub-agents, revise its own objectives, and chain together tool invocations in ways that neither the developer nor the human user could have fully anticipated.

Agentic AI represents the next, most volatile evolution of the non-human identity class. Unlike static software workloads, AI agents possess agency, autonomy, and persistent memory. They do not merely execute deterministic scripts; they use probabilistic reasoning to decompose high-level goals into sub-tasks, interacting with external environments through a Model Context Protocol (MCP) or specialised API tool sets, and with other agents through inter-agent (A2A) protocols.

The fundamental challenge in securing autonomous non-human identities lies in the *attribution gap* and the *intent–execution separation problem*. Traditional protocols assume that the client application is a faithful representative of the user's intent. In an agentic system, the LLM-based orchestrator sits between the user's command and the final execution. If an adversary manipulates the agent's reasoning path through prompt injection or memory poisoning, the agent may execute actions that are technically authorised by the client's credentials but represent a complete subversion of the user's original intent.

This chapter proceeds in three steps. Section 4.1 characterises the agentic threat landscape qualitatively. Sections 4.2 and 4.3 review traditional and agentic-specific threat-modelling frameworks, identifying the elements this thesis adopts. Section 4.4 introduces the proposed threat model, which synthesises elements of MAESTRO and ATFAA with a control-oriented severity-scoring requirement that the methodology in Chapter 5 fulfils.

4.1 Agentic Threat Landscape

The threat landscape for autonomous AI agents extends beyond transient prompt-level vulnerabilities into persistent, architectural exploit vectors [46]. As agents incorporate reasoning engines, external tools, and long-term memory, attackers increasingly target these structural elements rather than isolated model outputs. The attack surface spans the orchestrator (reasoning layer), the tool interface (execution layer), and the memory subsystem (persistence layer) [46].

Beyond industry taxonomies, academic research has proposed structured threat-modelling frameworks such as the Advanced Threat Framework for Autonomous AI Agents (ATFAA), which categorises risks across cognitive architecture vulnerabilities, temporal persistence threats, operational execution

manipulation, trust boundary violations, and governance circumvention [2]. ATFAA emphasises that threats such as reasoning-path hijacking, objective-function drift, and oversight saturation are not isolated events but systemic weaknesses emerging from autonomy and persistent state. Complementing this, the MAESTRO framework provides a layered architectural perspective for analysing vulnerabilities across the full AI stack, spanning foundation models, data operations, agent frameworks, infrastructure, observability, security controls, and the broader agent ecosystem [47]. Unlike traditional threat-modelling approaches, MAESTRO accounts explicitly for emergent behaviours arising from multi-agent interaction and stochastic reasoning, reinforcing the view that agentic threats are architectural rather than purely application-level [47].

The release of the OWASP Top 10 for Agentic Applications in December 2025 provides the first standardised taxonomy for autonomous AI security risks [48]. These risks reflect a shift from perimeter-based vulnerabilities toward logic-layer manipulation and identity exploitation. Among them, memory poisoning and cascading failures present the most recurrent architectural risks: they persist across sessions, exploit autonomy and memory, and cannot be addressed by static IAM.

Memory poisoning is one of the most insidious threat classes. Through indirect prompt injection, attackers implant malicious directives into an agent's long-term context, allowing compromises to remain dormant before activation [49]. The MINJA (Memory Injection Attack) lifecycle demonstrates how subtle injection, dormant storage, and later semantic triggering can produce delayed, high-impact compromise [49]. Similar research has shown that persistent prompt injection can corrupt session summaries, embedding malicious instructions that survive across future interactions [50]. Securing an agent's memory therefore requires *cognitive security*—treating an agent's recollections with the same scepticism applied to untrusted user input [49].

In multi-agent environments, stochastic reasoning chains amplify risk propagation. A compromised agent can influence downstream agents, enabling cascading failures that spread at machine speed; in simulated environments, a single compromised agent has been shown to poison 87% of downstream decision-making within four hours [46]. This phenomenon is compounded by stochastic sampling in LLMs, where a malicious prompt may succeed on one execution and fail on the next, making deterministic filtering extremely difficult [51]. The gradient of subtlety in modern injections—including invisible Unicode direction overrides—allows payloads to bypass heuristic filters while still succeeding in manipulating the model's instruction-following behaviour. The resulting threat landscape is persistent, adaptive, and architectural in nature.

4.2 Traditional Threat-Modelling Frameworks

Securing digital systems requires a systematic process to identify, enumerate, and prioritise threats. Traditional threat-modelling methodologies, developed over the last three decades, provided a robust framework for assessing vulnerabilities in software and networks. They were architected before the emergence of stochastic reasoning engines, however, and are primarily centred on static applications and fixed trust assumptions.

A common structural limitation across these methodologies is that they are designed for systems with relatively stable components, trust boundaries, and

principals—typically modelled through Data Flow Diagrams (DFDs), process flows, or organisational workshops. Tool-integrated LLM agents operate on untrusted natural-language inputs and may be manipulated, for example through prompt injection, into performing actions that exploit the agent's ambient authority and delegated access [52]. This mismatch is particularly pronounced for IAM in agentic AI, where agent identity is dynamic and characterised by a complex lifecycle involving creation, delegation, rotation, revocation, provenance, and audit. Delegation semantics—who is acting on whose behalf, with what scope, for how long, and under what constraints—are central to preventing confused-deputy behaviour and privilege escalation [53]. The six methodologies summarised below are each established within their original domain; each is described briefly, and the consolidated limitations for agentic systems are discussed at the end of the section.

4.2.1 STRIDE

STRIDE, developed by Loren Kohnfelder and Praerit Garg at Microsoft in 1999, is the most mature and widely adopted threat-modelling framework. It is an acronym for six threat categories—Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege—each corresponding to a core information-security principle [54]. The process involves building DFDs to identify entities, processes, data stores, and trust boundaries [54]. STRIDE's primary focus is on design and architecture, which can miss runtime-emergent threats; threat counts grow rapidly with system complexity, leading to a moderately high false-negative rate [54, 55].

4.2.2 PASTA

The Process for Attack Simulation and Threat Analysis (PASTA), introduced in 2012, is a risk-centric, seven-stage iterative process aligning risk modelling with business objectives [54]. The stages are: define objectives, define technical scope, application decomposition, threat analysis, vulnerability and weakness analysis, attack modelling, and risk and impact analysis [54]. PASTA is comprehensive and business-centric, integrating governance into security discussions from the outset, but is described as labour-intensive and difficult to scale in fast-paced development environments. While more realistic than STRIDE, it operates within relatively fixed system boundaries that may not be applicable to evolving agentic behaviour [55].

4.2.3 LINDDUN

LINDDUN, developed at KU Leuven, extends traditional security modelling to focus explicitly on privacy. The acronym stands for Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of Information, Unawareness, and Non-compliance [56]. It is widely recognised for its extensive privacy knowledge base and is well positioned to examine privacy-relevant attack surfaces in agentic AI such as prompts, tool outputs, retrieval contexts, logs, and memory [55]. However, LINDDUN is intentionally bounded to privacy concerns, whereas agentic-AI security incidents frequently span privacy, integrity, and authorisation simultaneously [57].

4.2.4 VAST

VAST is a scalable, automatable threat-modelling methodology suited to agile development environments. It supports the SDLC through two complementary perspectives: an operational threat model, which applies an attacker-oriented DFD view to infrastructure, and an application threat model, which uses process-flow mapping to capture developer-facing functionality and communication [58]. This separation is advantageous in agentic-AI programmes, where architectures span application logic, orchestration layers, and operational toolchains [58].

4.2.5 OCTAVE

OCTAVE is an organisationally grounded risk-assessment methodology defining self-directed activities for identifying and managing information-security risks. It concentrates on mission-critical assets, associated threats and vulnerabilities, and protection strategies aligned with organisational risk tolerances [59]. Its principal strength is governance: it ensures that threats are analysed within their broader organisational context, including people, technology, processes, and facilities [59]. These characteristics remain relevant for agentic-AI deployments, where risk is inherently socio-technical and extends beyond software components [58].

4.2.6 Trike

Trike is a risk-management-oriented security-auditing framework that applies threat modelling to generate reliable and repeatable results. It explicitly separates a requirements model—comprising actors, assets, intended actions, and an actor–asset–action matrix—from an implementation model that includes DFDs and use flows, with the objective of supporting partial automation and ensuring consistent communication across stakeholders [60]. The actor–asset–action representation is conceptually well aligned with agentic AI, since such systems require a clear specification of what actions an agent is expected to perform and which assets those actions may affect [60].

4.2.7 Common Limitations for Agentic Systems

These six methodologies share a structural blind spot that is particularly consequential for agentic AI. Their underlying representations—DFDs, process flows, actor–asset–action matrices, and organisational workshops—are designed for systems whose principals are deterministic and whose trust boundaries are stable. They under-represent the core agent loop (reasoning, tool selection, action, memory update) and the fact that, in tool-integrated LLM applications, natural-language inputs can function as control data [55]. This is precisely the condition exploited by direct and indirect prompt injection.

From an IAM perspective, these methodologies' categories of spoofing and elevation of privilege remain useful but are too coarse to capture agentic security challenges, where the dominant hazards arise from delegation and ambient authority rather than simple endpoint impersonation. The confused-deputy pattern—in which a component possessing its own authority is induced to misuse that authority on behalf of another party—is a classic access-control failure mode that becomes especially

prevalent when an agent holds broad API tokens and is prompted to invoke tools [61]. The standard workflows of these frameworks do not natively encode formal acts-for or on-behalf-of relationships, delegation scope, or multi-principal policy composition, despite the fact that these are well-established foundations in distributed access-control logics [62]. In practice, this creates gaps in modelling: (a) credential-lifecycle management, including issuance, rotation, revocation, and binding; (b) multi-agent authorisation, including agent-to-agent calls and chained tool execution; and (c) identity provenance, namely how a given agent identity is established and verified across sessions and tool boundaries [63, 64, 57]. A further empirical observation is that strong procedural guidance can constrain analyst creativity: in evaluation settings, participants often fail to identify threats that fall outside the provided catalogue [65], which becomes a structural concern in agentic AI where attack techniques and IAM abuse patterns evolve rapidly.

Table 4.2.1: Comparison of Traditional Threat Modeling Methods in the Context of Agentic AI

Methodology	Scope	AI-agent-specific gaps	IAM-specific gaps for agentic AI
STRIDE	Software- and system-centric security analysis (DFD-based)	Limited treatment of instruction channels, memory, and tool-mediated autonomy; weak coverage of indirect prompt injection and related control-flow manipulation	Lacks native support for delegation semantics, scoped authority, credential lifecycle control, and identity provenance across chained agents
PASTA	Risk-centric analysis using abuse cases and attack modeling	Difficult to represent non-deterministic agent behavior and untrusted natural-language inputs as part of the control surface	Delegation, token binding, and authority composition are often implicit, which may obscure authorization drift in multi-agent settings
LINDDUN	Privacy threat modeling based on DFDs and privacy threat trees	Strong privacy orientation may underrepresent emergent agent security issues, particularly those involving integrity, authorization, and tool use	Emphasizes identity as a privacy concern rather than as an authentication and authorization construct; limited support for delegation and agent identity lifecycle modeling
VAST	Enterprise-scale, agile-oriented threat modeling for operational and application contexts	Remains largely architecture-centric and offers limited abstraction for instruction-boundary attacks, reasoning-state exposure, and memory-related threats	Provides limited formalization of "on-behalf-of" actions, scoped delegation, and credential governance in agentic environments
OCTAVE	Organizational risk assessment focused on critical assets and workshops	Its high-level perspective may overlook concrete agent-level threats such as indirect prompt injection and manipulation through tool-integrated workflows	Tends to treat agent identity and delegation as implementation details, with limited explicit modeling of delegated authority and multi-agent authorization chains
Trike	Risk-based security auditing with requirements and implementation models	Its action-centered structure is less stable for LLM agents whose behavior is adaptive, planning-driven, and influenced by untrusted content	Offers a useful basis for access modeling, but remains incomplete without explicit treatment of delegation, provenance, revocation, and policy composition

4.3 Agentic-Specific Threat-Modelling Frameworks

In contrast to the traditional methodologies above, several frameworks have been developed specifically for agentic AI. They fall into two groups: taxonomies and scoping matrices that catalogue threats, and structured frameworks that provide architectural decomposition and threat-localisation methodology. This thesis adopts elements of the structured frameworks and treats the taxonomies as inputs.

4.3.1 Threat Taxonomies and Scoping Matrices

Agentic-AI security guidance has rapidly proliferated in the form of threat taxonomies and scoping matrices, including the OWASP Top 10 for Agentic Applications 2026 [2] and the AWS Agentic AI Security Scoping Matrix [28]. These artefacts are valuable because they standardise vocabulary, highlight high-impact failure modes, and provide an initial prioritisation compass for practitioners.
As inputs to threat modelling rather than substitutes for it, however, they have structural limitations. Taxonomies and matrices are vector-centric: they list or classify attack types, but do not force a structured decomposition of the deployed agentic system, its trust boundaries, its agent/tool/identity relationships, or the lifecycle of agent authorities. The OWASP Top 10 enumerates agent-specific threats such as goal hijack, tool misuse, identity and privilege abuse, insecure inter-agent communication, and memory poisoning, which is essential for coverage; but the Top 10 format does not produce system-specific threat traces that connect where a threat enters, how it propagates across planning, memory, and tool boundaries, and which IAM control plane must enforce constraints at each step [2]. The AWS Scoping Matrix classifies agentic use cases into four scopes based on agency and oversight [28]; this is useful for posture and control-maturity scoping by autonomy level but is not by itself a threat model: it does not enumerate trust boundaries, derive attack chains, or provide explicit threat-to-architecture mappings [28].
Beyond these two, other matrix-like resources are commonly used to orient security analysis. MITRE ATLAS positions adversarial-ML attacks in an ATT&CK-style matrix to help analysts orient themselves to ML threats [66, 47]. NIST AI 100-2e2023 provides a standardised taxonomy and terminology of adversarial-ML attacks and mitigations [67]. The NIST AI Risk Management Framework 1.0 is governance-oriented and intended to help organisations manage AI risks across lifecycle stages, providing a risk-management structure rather than an operationalised agentic threat model [68]. These resources are highly relevant for AI security coverage but, like the OWASP Top 10 and AWS matrix, remain knowledge bases or taxonomies rather than complete threat-modelling workflows that produce architecture-specific artefacts [47].
This thesis treats taxonomies and matrices as inputs and validation checklists, primarily to define attack vectors and the system state, while selecting agentic threat-modelling frameworks that (i) structure analysis around agent architecture and emergent behaviours, and (ii) surface IAM requirements central to agent autonomy.

4.3.2 MAESTRO

The MAESTRO framework is a multilayer threat-modelling architecture for agentic systems that decomposes the operational stack into seven connected layers to localise threats and support mitigation at the appropriate layer [55]:

- **L1 Foundation Models**
- **L2 Data Operations** (pipelines, labelling, storage)
- **L3 Agent Frameworks** (orchestration and decision-making between agents)
- **L4 Deployment and Infrastructure** (container/cloud hosting)
- **L5 Evaluation and Observability** (monitoring and integrity)

- **L6 Security and Compliance** (privacy, access control, regulatory assurance)
- **L7 Agent Ecosystem** (multi-agent collaboration and interaction with external items)

This layered decomposition addresses a key weakness of traditional approaches: conventional methodologies often assume static components and stable trust boundaries, whereas agentic systems require explicit attention to memory, planning, and tool invocation as first-class attack surfaces [47]. MAESTRO's most relevant structural contribution for IAM is that it forces placement of identity and control questions into Layer 6 while still enabling cross-layer propagation analysis—for example, prompt or input manipulation in L7 or L3 leading to unauthorised tool behaviours that must be constrained by IAM policy enforcement at L6 [47]. This creates a stable anchor for the question "where do IAM controls live?" even when agent behaviour is dynamic.

4.3.3 ATFAA

The Advanced Threat Framework for Autonomous AI Agents (ATFAA) is a threat model tailored to generative AI agents that "reason, remember, and act," often embedded in enterprise systems with minimal human oversight. ATFAA explicitly argues that many existing frameworks treat LLMs as isolated components and therefore do not capture the emergent security properties created by autonomy, long-term memory, and dynamic tool use [69]. It organises agentic threats into five domains and identifies nine primary threats, mapped to STRIDE categories and paired with a mitigation framework (SHIELD) [69].

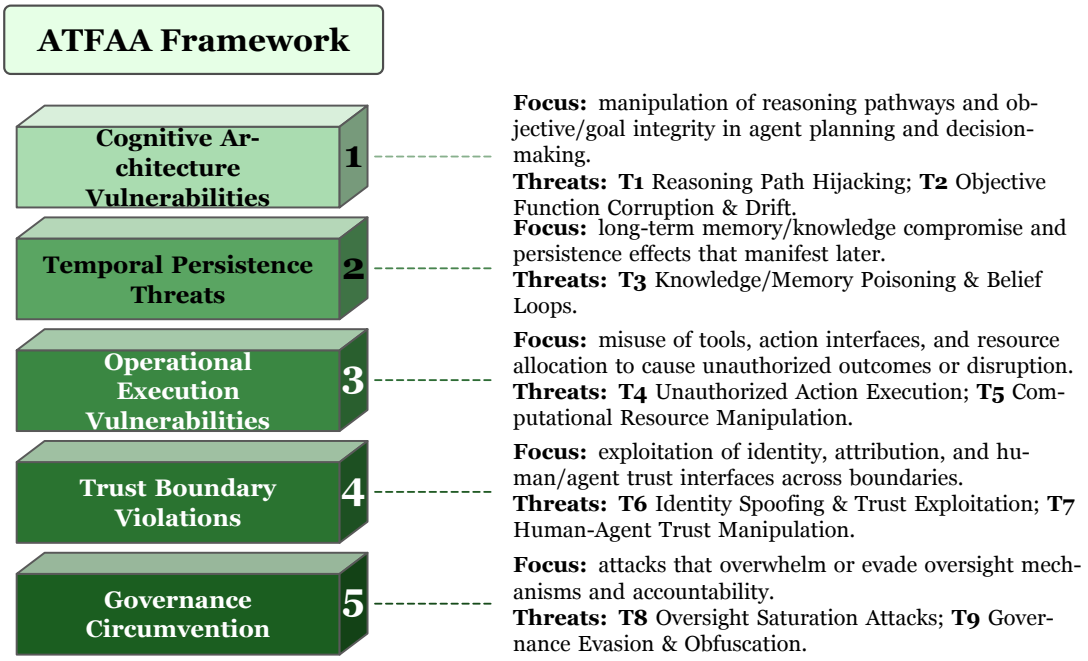

Figure 4.3.1: ATFAA domains and mapped threats

ATFAA provides unusually direct coverage of agentic IAM dynamics in two ways.

First, it elevates IAM as a first-class threat surface via T6 (Identity Spoofing and Trust Exploitation), which targets weaknesses in identity verification and "identity transition zones" where agent actions become attributed to human identities, as well as inter-agent trust attestation channels [69]. This aligns with the enterprise reality that many agent compromises manifest not as pure model failures but as authorisation-context corruption. Second, ATFAA explicitly characterises the *identity fluidity* problem: the blurry line between user identity and agent identity creates new impersonation and privilege-escalation opportunities that challenge traditional authentication and authorisation models [69].
ATFAA has limitations that motivate how it is extended in this thesis. First, ATFAA itself calls for future empirical validation and the development of quantitative risk-assessment methodologies tailored to agentic threats, confirming that its threat coverage is not yet fully ground-truthed and that its risk-scoring approach remains primarily qualitative [69]. Second, ATFAA is a threat taxonomy and does not, by itself, provide a detailed layered reference architecture comparable to MAESTRO; applying ATFAA in practice can therefore benefit from architectural decomposition to ensure coverage across model, memory, tooling, infrastructure, and ecosystem layers [69].
These limitations motivate the combined approach this thesis adopts: ATFAA provides the threat-model foundation, while MAESTRO is used as a layer-mapping scaffold to ensure architectural completeness and traceability.

4.4 Proposed Threat Model

The proposed threat model is grounded in a systematic decomposition of the agentic ecosystem into the seven MAESTRO layers [47]. This layered approach ensures that the analysis of assets, adversaries, and attack surfaces is not confined to isolated software components but accounts for the complex dependencies inherent in autonomous systems [47]. The architectural scope encompasses Foundation Models (L1), Data Operations (L2), Agent Frameworks (L3), Deployment and Infrastructure (L4), Evaluation and Observability (L5), Security and Compliance (L6), and the Agent Ecosystem (L7) [47].
By adopting this seven-layer architecture, the threat model provides a mapping scaffold that links technical vulnerabilities to operational impact across the "reason, remember, act" lifecycle [47]. The subsequent subsections define seven analytical dimensions: protected assets, adversary profile, attack vectors and mechanisms, attack surface, attack methods, trust boundaries, and failure propagation paths. The model follows an analytical path: identifying *what* to protect (assets), *who* might attack (adversaries), *how* they enter (surface and vectors), *where* the intent is lost (trust boundaries), and *how far* the damage spreads (propagation) [70, 55]. A separate section then discusses risk assessment and severity scoring as a requirement that the methodology in Chapter 5 fulfils.

4.4.1 Assets

The threat model adopts an asset-centric perspective, identifying the critical components whose compromise could enable adversarial manipulation of autonomous agent behaviour [70, 69].
At the cognitive level, assets include the foundation-model weights, reasoning logic,

and system prompts that govern decision-making; corruption at this level may lead to reasoning subversion or model inversion [70]. Closely related are contextual assets such as long-term memory stores, session summaries, and retrieval-augmented generation corpora, which represent persistent state and are primary targets for delayed or long-horizon poisoning attacks [2]. Identity-related assets form a further critical category, encompassing cryptographic identifiers, delegation tokens, and proof-of-possession keys that bind agent actions to authorised intent; compromise of these assets threatens traceability, accountability, and trust enforcement across workflows [22]. Execution infrastructure—secure runtimes, containers, hardware-backed keys, and trusted execution environments—provides the computational foundation upon which agent behaviour is executed and protected [70]. Operational assets include external tool interfaces, APIs, and privileged execution capabilities through which agents interact with external systems; since agents can autonomously invoke tools, these interfaces significantly expand the attack surface and require strict governance. Finally, governance telemetry—behavioural logs, reasoning traces, and operational metadata—constitutes a critical defensive asset, as it enables forensic analysis, anomaly detection, and continuous verification [69].

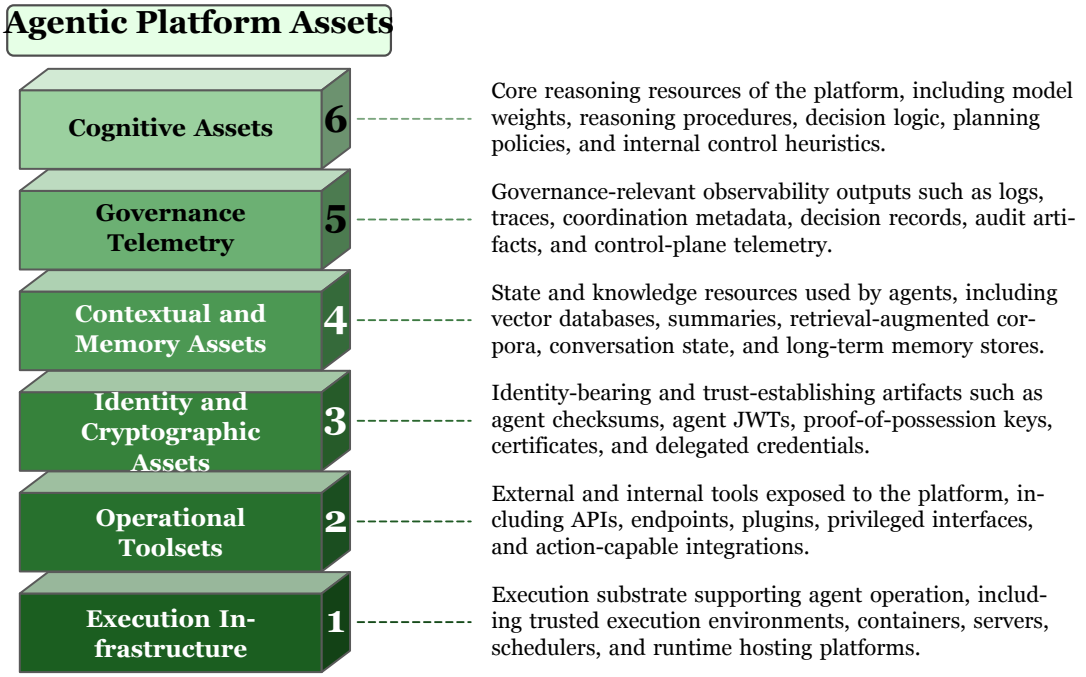

Figure 4.4.1: Six protected asset categories in the agentic platform (asset-centric threat-modelling view).

4.4.2 Adversary Profile

Adversaries are characterised by knowledge level, access capabilities, and operational objectives. White-box adversaries possess full visibility into the system's internal components, including model weights, prompts, tool configurations, and execution logic. Such adversaries typically emerge from insider threats or leaked system artefacts and aim at precise reasoning hijacking, optimal evasion of defences, or targeted manipulation of agent behaviour [70]. Gray-box adversaries operate with

partial knowledge—typically from metadata exposure, inference APIs, or indirect access to retrieval pipelines—and exploit structural understanding of data flows to conduct targeted poisoning, model inversion, or context-accumulation manipulation [70]. Black-box adversaries have no internal visibility and rely on observable behaviour through public or user-facing interfaces, employing indirect prompt injection, social engineering, or iterative probing to influence agent decision-making; despite limited visibility, they remain effective due to the probabilistic nature of agent reasoning and the difficulty of distinguishing malicious instructions from legitimate inputs [70].
Across all categories, adversarial objectives extend beyond traditional data theft toward subverting institutional trust and manipulating agent intent while remaining within technically authorised execution boundaries, thereby exploiting the attribution gap inherent to autonomous systems.

4.4.3 Attack Vectors and Mechanisms

Agentic systems introduce a diverse set of attack vectors that target the cognitive, contextual, and operational layers of autonomous agents. These vectors include reasoning-path manipulation through indirect prompt injection, memory and context poisoning, cascading failures in multi-agent environments, parameter pollution, action chaining, inference-based model extraction, adversarial evasion, tool manipulation, and supply-chain or infrastructure-level compromise [2]. Collectively, these vectors illustrate a shift from transient software vulnerabilities toward persistent, architectural exploitation of agent autonomy and identity boundaries.
While the complete attack surface is broad, this work focuses on three representative attack vectors: goal hijacking (ASI01), memory poisoning (ASI06), and cascading failures (ASI08) [2]. These were selected because they represent structural risks that persist across sessions, exploit stochastic reasoning and autonomy, and directly expose limitations in static IAM [70]. Goal hijacking targets the cognitive layer by manipulating reasoning trajectories; memory poisoning exploits persistent contextual storage to introduce delayed malicious behaviour; and cascading failures demonstrate how compromise propagates across collaborative agent networks at machine speed.
By focusing on these three vectors, the proposed framework addresses the core behavioural mechanisms underlying many agentic threats. The combination of behavioural anomaly detection and adaptive privilege control enables runtime mitigation of reasoning deviation, unsafe tool invocation, and uncontrolled agent propagation. Although not every underlying vector is explicitly prevented, this behavioural governance approach mitigates a substantial portion of the OWASP Top 10 for Agentic Applications at the execution level by continuously monitoring agent behaviour and dynamically constraining privileges based on observed risk.

Scope of Attacks in the Threat Model versus the Experimental Dataset

A deliberate scope asymmetry exists between the threat model and the experimental dataset in Chapter 6, and it should be made explicit. Chapter 4 focuses on three *structural* attack classes—goal hijacking, memory poisoning, and cascading failures—because these represent the fundamental ways in which an agent's behaviour can diverge from authorised intent while its credentials remain valid. They are chosen not because they exhaust the agentic threat surface, but because they

directly exercise the assumptions that classical IAM cannot enforce: stochastic reasoning, persistent context, and autonomous delegation. The proposed behaviour-driven framework is designed to address these three classes at the mechanism level, independent of the surface form of the attack.
The experimental dataset, however, contains a broader set of attack variants drawn from the OWASP Top 10 for LLM Applications, the OWASP Top 10 for Agentic Applications, and supplementary categories covering network and physical-layer perturbations. This broader set serves three purposes. First, it provides concrete instantiations of the three structural classes: goal hijacking is realised through direct prompt injection (`attack_inject_skip`), orchestration hijacking (`attack_hijack`), and excessive-agency attacks (`attack_admin_delete`); memory poisoning through indirect prompt injection via observation payloads (`attack_indirect`) and inter-agent manipulation (`attack_manipulate`); and cascading failures through oscillation (`attack_oscillation`), denial-of-service loops (`attack_dos_loop`), and logic-skipping attacks (`attack_skip_node`). Without multiple concrete instantiations of each structural class, the evaluation could not distinguish whether the framework generalises across surface forms or merely memorises a single attack signature.
Second, the broader set functions as a set of out-of-distribution robustness probes. Attacks such as information leakage (`attack_leakage`), context padding (`attack_dos_padding`), sensor spoofing (`attack_spoof_low`), and network latency injection (`attack_latency`) do not correspond directly to any of the three structural classes. Including them allows measurement of the framework's behaviour on adversarial inputs that it was not explicitly trained or designed to catch: if the behavioural detector flags them, the learned representation captures a broader notion of deviation than the three target classes alone; if it does not, the result quantifies the framework's generalisation boundary.
Third, the broader set enables meaningful false-positive and utility analysis. A detector that responds only to the three target classes could achieve high detection rates by over-flagging unusual behaviour. Including adversarial scenarios outside the design envelope, together with diverse normal scenarios, allows the evaluation to distinguish genuine behavioural signal from indiscriminate anomaly sensitivity. This is reinforced by the dataset's regime structure: attack episodes are graded into successful, partial, and blocked outcomes, producing a four-regime evaluation space (normal, successful, partial, blocked) in which the framework's discrimination ability can be measured at finer granularity than a binary normal-versus-attack split would allow. In summary, the three structural classes define what the framework is *designed to address*; the broader attack set defines what it is *evaluated against*.

4.4.4 Attack Surface

The attack surface of autonomous AI agents expands significantly compared to traditional software systems due to their ability to reason, maintain persistent state, and autonomously interact with external tools and other agents. Exposure is not limited to external network interfaces but includes internal cognitive and execution pathways where decisions are formed and actions are delegated.
The first surface consists of input channels: user prompts, retrieved documents, web pages, and other external data sources that enter the agent's contextual state. Because agents continuously ingest untrusted information, these interfaces are primary entry

points for indirect prompt injection and contextual manipulation [70]. A second surface emerges within the planning and reasoning layer, where the orchestrator decomposes objectives and determines execution strategies; manipulation at this stage can redirect goals without violating explicit authorisation constraints [47]. A third surface appears at the tool-execution boundary, where agents invoke external APIs, system commands, or enterprise services; here, inherited credentials and implicit trust relationships introduce risks such as parameter pollution and unauthorised tool usage [71]. Persistent memory systems form a fourth surface, as long-term storage allows malicious context to remain latent and influence future reasoning [47]. Finally, inter-agent communication constitutes a fifth surface, where compromised agents can propagate malicious instructions or false consensus to downstream collaborators [72].

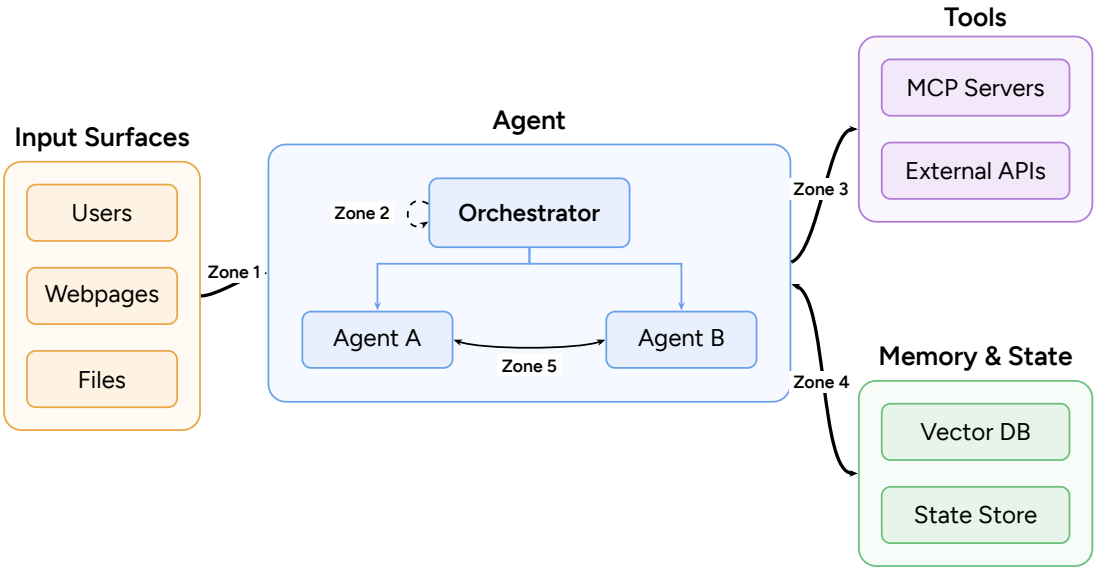

Figure 4.4.2: Attack surface decomposition across five zones in an agentic system architecture.

4.4.5 Attack Methods

The taxonomy of adversarial techniques is bifurcated into foundational machine-learning vulnerabilities and agent-specific operational risks [66]. The foundational methods—evasion, poisoning, and inference—are derived from the NIST AI 100-2 standard, the authoritative baseline for adversarial machine learning [27]. Evasion exploits inference-time vulnerabilities through adversarial perturbations or linguistic manipulations to bypass safety guardrails; poisoning corrupts the underlying logic through training data or retrieved context; inference attacks target the extraction of proprietary model weights or sensitive data through systematic API querying [70].

To address the specialised challenges introduced by autonomous entities, the framework also incorporates operational methods identified in the OWASP Top 10 for Agentic Applications and ATFAA. Action chaining and rug-pull attacks exploit the agency gap where probabilistic reasoning loops interact with distributed toolsets [2]. Action chaining strategically sequences individually benign operations to achieve an unauthorised aggregate outcome, such as exfiltration through encoded parameters

[2]. Rug-pull attacks address supply-chain vulnerabilities where the metadata or functional behaviour of an approved tool or prompt template is mutated post-authorisation to execute malicious instructions [2].

4.4.6 Trust Boundaries

Trust boundaries define the points within an agentic system where assumptions about intent, authority, or execution integrity change, and where Zero-Trust controls must be explicitly enforced. This is one of the main reasons that classical IAM fails to achieve the principle of least privilege or Zero-Trust principles within agentic systems: there is an inevitable gap between resource-owner intent and the actions taken by the agent acting on its behalf.

The first boundary exists between humans and agents, where user instructions are translated into internal goals and reasoning trajectories. Misalignment at this stage leads to intent–execution separation, allowing an agent to perform technically authorised actions that no longer reflect the user's original objective [2]. A second boundary appears between collaborating agents in multi-agent systems, where delegation and message exchange introduce risks of impersonation, collusion, or malicious context propagation [72]. A third boundary lies between the infrastructure and the model runtime, where secure execution environments must prevent host-level compromise from exposing cryptographic assets or altering agent behaviour [70]. Finally, the boundary between agents and external tools represents a critical point of trust transfer, as agents inherit permissions to interact with APIs and enterprise systems, creating potential confused-deputy scenarios if execution is not continuously verified [2].

Identifying and enforcing these trust boundaries is essential for restoring Zero-Trust principles in agentic ecosystems, ensuring that intent, identity, and execution remain continuously verified rather than implicitly trusted.

4.4.7 Failure Propagation Paths

Identifying failure-propagation paths is essential for managing the systemic risks inherent in autonomous multi-agent systems, where security incidents can cascade at machine speed through complex delegation chains [72]. Unlike traditional software, where vulnerabilities are often contained within isolated process or memory boundaries, agentic compromises tend to move fluidly across architectural layers and trust zones [47]. A primary propagation vector is the exploitation of trust relationships between specialised agents [55]: a single compromised upstream agent can contaminate the reasoning context of the entire network through false consensus messages or malicious instruction propagation, a phenomenon categorised as Cascading Failures (ASI08) [55, 2].

Propagation can be analysed through cross-layer threat identification, tracing how a minor vulnerability in one MAESTRO layer is leveraged to create catastrophic failure in another [47]. For example, a latent prompt-injection trigger at the Foundation Model layer (L1) can propagate through planning logic in the Agent Framework (L3) to manifest as unauthorised tool execution in the Agent Ecosystem (L7) [47]. Spatial propagation occurs across concurrently running agents; temporal propagation occurs when poisoned contextual data remains dormant in memory (L2) until specific semantic triggers activate a high-impact malicious workflow in future sessions. With

current trends toward long context windows in agentic applications, tracking long-persistent indirect prompt injections placed in long-term memory can be extremely hard; this is discussed further in Chapter 8.
Detailed analysis indicates that propagation is often driven by state-synchronisation failures: stale-state propagation, where an agent operates on outdated information because a peer's update was not received in time, and conflicting state updates, where concurrent modifications to shared state corrupt logical consistency [55]. Using the five-zone lens—input, reasoning, tool execution, memory, and communication—the threat model identifies critical bifurcation points where an adversary can redirect the agent's goal trajectory across multiple execution cycles. This structural analysis informs the placement of circuit-breaker mechanisms and determines where the Model Predictive Control logic in the methodology must enforce graduated containment to prevent system-wide instability.

4.4.8 Risk Assessment and Severity Scoring

A threat model for autonomous agents is incomplete without a corresponding mechanism for quantifying the severity of observed behaviour. Classical IAM systems do not require runtime severity scoring because authorisation decisions are binary and made once per session. In agentic systems, however, risk evolves continuously as the agent reasons, invokes tools, and interacts with external resources, and severity must therefore be expressed as a time-varying quantity that can drive adaptive responses [70, 69].
Existing AI governance frameworks, including the NIST AI Risk Management Framework [27] and scoring rubrics such as the OWASP AI Vulnerability Scoring System (AIVSS) [72], provide valuable qualitative structure but remain primarily descriptive. They are intended for pre-deployment risk review and organisational governance rather than for runtime intervention at machine speed [69]. In production environments where agents execute multi-step plans across delegated sub-agents and external APIs, manual point-in-time assessments are impractical, and a quantitative, continuously computed severity signal is required [69].
Any severity-scoring mechanism suitable for closed-loop IAM must therefore satisfy three properties. First, it must be *continuous and online*, producing an updated score at every decision step rather than only at session boundaries. Second, it must be *behaviourally grounded*, deriving its signal from observed agent trajectories rather than from static attributes or credentials, so that the same identity can be associated with different severity levels at different times depending on how it behaves. Third, it must be *calibrated*, in the sense that its numerical output can be mapped consistently onto operational severity levels that downstream controllers can act upon.
This thesis treats the threat model as defining the requirements for such a scoring mechanism and defers its concrete instantiation to Chapter 5, where the instantaneous severity signal is derived from the reconstruction behaviour of a CVAE+LSTM model trained on normal agent trajectories and is combined with a POMDP-based historical trust estimate to form the state vector used by the Model Predictive Control access controller. The threat model's role here is not to specify the algorithm, but to establish that severity scoring is a first-class requirement of agentic IAM and that its output must feed directly into the access decision process.

4.5 Threat Model Summary

The overall threat model assumes that autonomous AI agents are not merely software clients, but stochastic, identity-bearing actors whose reasoning, memory, and tool use can be subverted while still operating under technically valid credentials. The central risk is therefore the attribution gap and the intent–execution separation problem: an agent may execute an action that is formally authorised by the client's credentials but no longer reflects the original user intent. To capture this, the model adopts an asset-centric, bottom-up perspective and identifies six protected asset classes within the agentic platform: cognitive assets, contextual and memory assets, identity and cryptographic assets, execution infrastructure, operational toolsets, and governance telemetry. This framing reflects the fact that attacks against agentic systems are typically architectural and cross-layer, rather than isolated software bugs.
On this basis, the threat model characterises adversaries by knowledge level and access capability, ranging from black-box prompt injectors to gray-box data-flow manipulators and white-box insiders with visibility into prompts, tools, and execution logic. It maps the main attack vectors to the core trust boundaries of the system, with particular emphasis on goal hijacking, memory poisoning, and cascading failures, since these threats exploit persistent state, stochastic reasoning, and autonomous tool invocation in ways that static IAM cannot adequately contain. The summary conclusion is that securing agentic systems requires continuous, behaviour-aware governance across the full stack, which directly motivates the methodology developed in the next chapter.

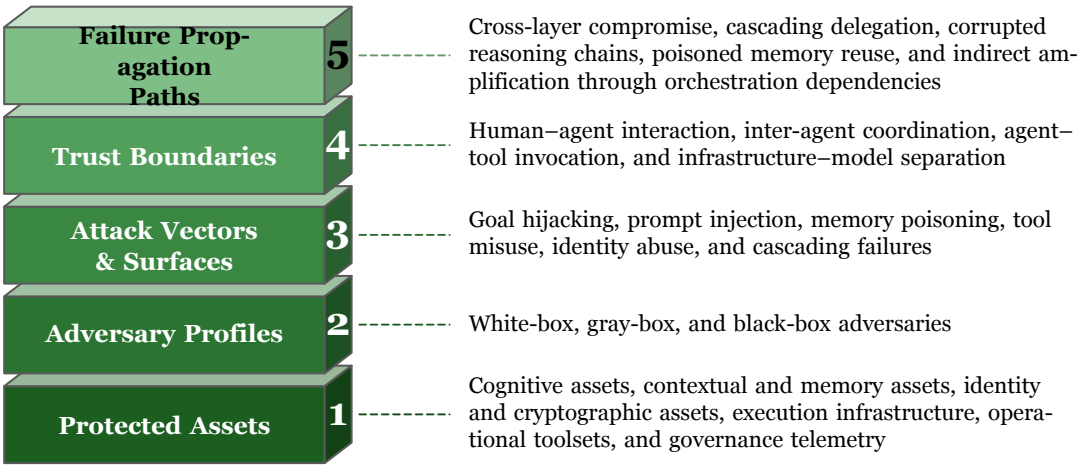

Figure 4.5.1: Suggested threat model

5 Proposed Methodological Framework

This chapter presents the general methodological framework proposed for behaviour-aware Identity and Access Management in autonomous agentic systems. It defines the target architecture and the mathematical relationships between behavioural risk estimation, historical trust, predictive access control, and hard safety constraints. The framework is therefore broader than the exact experimental implementation evaluated in Chapters 6 and 7.

Because of limitations in the available simulation environment, dataset, computational resources, and project scope, not every component of the proposed framework could be implemented in its full form. The experiments consequently use simplified instantiations of the proposed modules while preserving the main information flow:

$$\text{behavioural observations} \rightarrow \text{risk} \rightarrow \text{historical trust} \rightarrow \text{adaptive access decision}. \tag{5.0.1}$$

The exact code-level implementation and its mathematical formulation are documented in Appendix A.

5.1 Overview

We consider an agentic AI system where each agent i is associated with a static identity $\mathrm{ID}_i \in \mathcal{I}$ and a time-varying security state derived from behavioral observations.

At time t, the agent requests a set of actions $\mathcal{A}_{i,t}^{\mathrm{req}}$. The system computes a dynamic claim vector

$$\xi_{i,t} = [r_{i,t},\ \tau_{i,t},\ c_{i,t},\ q_{i,t}], \tag{5.1.1}$$

where:

- $r_{i,t} \in [0,1]$ is instantaneous risk,
- $\tau_{i,t} \in [0,1]$ is historical trust,
- $c_{i,t}$ is contextual information,
- $q_{i,t}$ is action criticality.

The runtime principal is therefore

$$\mathrm{Principal}_{i,t} = (\mathrm{ID}_i,\ \xi_{i,t}). \tag{5.1.2}$$

The goal is to compute a control decision consisting of:

$$u_{i,t} = (h_{i,t}, m_{i,t}), \tag{5.1.3}$$

where:

- $h_{i,t} \in \mathcal{H}$ is the autonomy level,
- $m_{i,t} \in \{0,1\}^{N_t}$ is the action permission vector.

Latent behavioural clustering is considered as a theoretical extension of this state representation. In the experimental implementation, no explicit cluster assignment is supplied to the trust estimator or access controller; the downstream components use the calibrated reconstruction-based risk signal $r_{i,t}$ instead.

5.2 Behavioral Modeling via a Conditional Recurrent Latent-Variable Model (CVAE+LSTM)

We model agent behavior with a conditional sequential latent-variable model, and motivate the choice from the inference problem it must solve, rather than from the architecture itself. The threat model (Chapter 4) characterizes a compromised agent by a hidden security state—faithful versus subverted—that is never observed directly and can only be inferred from behavioral emissions. Estimating this state online is therefore a Bayesian filtering problem: we require the posterior over a latent state given the history of observations. The exact filter (the POMDP belief update of Section 5.3, following [73]) presupposes an observation model over a high-dimensional, multimodal, nonlinear behavioral space that cannot be specified by hand. The variational autoencoder learns exactly such an intractable likelihood model from data, as amortized variational Bayesian inference [74]; this is the reason the behavioral model is built on it. Three properties of the problem then fix the architecture.

Generative and one-class, hence a calibrated likelihood-based risk. Attacks are rare, diverse, and partly unseen at training time (Chapter 1), so we model the distribution of *normal* behavior and score deviations by how poorly the model explains new behavior, rather than training a discriminative classifier that would require labeled attacks. The VAE provides this as a probabilistic encoder–decoder trained to maximize a variational lower bound on the marginal log-likelihood [74]; scoring anomalies by the model's reconstruction probability [75] yields a continuous, calibratable signal, which is precisely the scoring-function-not-binary-label requirement of behavioral risk evaluation [44] and the continuous, online, calibrated severity signal required by the threat model (Section 4.4.8) [69]. A learnable Gaussian-mixture prior is also evaluated (Section 7.3.1) to examine whether a multimodal latent distribution better represents distinct modes of normal behaviour (Section 5.2.2). The mixture components may provide an interpretable basis for latent behavioural clustering [76]; however, explicit component assignments are not used by the trust estimator or controller in the experimental implementation.

Recurrent, hence history-conditioned normality. The attacks of interest (e.g. goal hijacking, memory poisoning, cascading failures) are defined by trajectories, not single requests: a poisoned memory remains latent and shapes future steps, and a successful attack can appear benign step by step while being adversarial in where the trajectory ends. A memoryless model cannot, in principle, detect such drift. We therefore condition the latent model on a recurrent summary of history $h_{i,t-1}$ produced by an LSTM [77], so that the encoding and decoding describe what is

normal given the trajectory so far (meaning history-conditioned normality rather than static normality). Conditioning a VAE on a recurrent state in this way is exactly the variational RNN (VRNN) construction [78], a conditional generative model [79] in which the conditioning variable is the recurrent state; we adopt this construction and refer to it as CVAE+LSTM. The same VAE-with-recurrence pattern has been used to turn behavioral and sensor streams into continuous risk and anomaly scores in adjacent domains [80] [44], which supports the choice empirically.

Let $x_{i,t} \in \mathbb{R}^d$ denote the observed behavioral feature vector. Temporal context is modeled via an LSTM,

$$h_{i,t} = \mathbf{LSTM}(x_{i,t},\, h_{i,t-1}), \qquad h_{i,t} \in \mathbb{R}^m, \tag{5.2.1}$$

where $h_{i,t}$ is the recurrent hidden state. The CVAE conditions the latent encoding and decoding on this temporal state, so the model learns history-conditioned normality:

$$q_\phi(z_{i,t} \mid x_{i,t},\, h_{i,t-1})\,, \qquad p_\theta(x_{i,t} \mid z_{i,t},\, h_{i,t-1})\,. \tag{5.2.2}$$

The training objective is the conditional evidence lower bound (ELBO),

$$\mathcal{L}_t = \mathbb{E}_{q_\phi}\big[\log p_\theta(x_{i,t} \mid z_{i,t},\, h_{i,t-1})\big] - D_{\mathrm{KL}}(q_\phi(z_{i,t} \mid x_{i,t},\, h_{i,t-1}) \,\|p(z))\,, \tag{5.2.3}$$

and for a full sequence of length T,

$$\mathcal{L}(\phi, \theta) = \sum_{t=1}^{T} \mathcal{L}_t(\phi, \theta). \tag{5.2.4}$$

Why this model rather than the alternatives. A plain autoencoder or a discriminative classifier yields no calibrated likelihood and requires labeled attacks; a hidden Markov or linear-Gaussian filter is the correct Bayesian-filtering ancestor but cannot represent a high-dimensional, multimodal, nonlinear behavioral emission (the VRNN is the deep nonlinear realization of the same idea [78]); a GAN provides no encoder and no tractable likelihood and is unstable on sequential data; a deterministic next-step predictor returns a point estimate rather than the posterior the belief filter consumes. Among models that simultaneously provide (i) a likelihood-based calibrated score, (ii) a latent representation capable of modelling multimodal behavioural structure, (iii) native temporal conditioning, and (iv) cheap online inference, the conditional recurrent latent-variable model is the one that satisfies all four (which is what allows the POMDP trust filter (Section 5.3) and the MPC controller (section 5.4.3, [81]) to be layered on top of it) .

Link to the trust filter. The CVAE–LSTM converts each behavioural window into a reconstruction error, which is calibrated to obtain the instantaneous risk signal $r_{i,t}$. In the experimental implementation, this risk signal is discretized into observation categories and supplied to the POMDP belief update. The latent posterior and possible mixture-component assignments are analysed as learned representations but are not direct inputs to the trust filter.

The complete implemented network architecture and tensor dimensions are provided in Appendix A, Figure A.3.1.

5.2.1 Instantaneous Risk Estimation

Reconstruction error is defined as

$$e_{i,t} = \ell(x_{i,t}, \hat{x}_{i,t}), \quad \hat{x}_{i,t} \sim p_\theta(\cdot \mid z_{i,t}, h_{i,t-1}). \tag{5.2.5}$$

Risk is obtained via calibration:

$$r_{i,t} = \varphi(e_{i,t}) \in [0, 1], \tag{5.2.6}$$

where φ is a normalization or probabilistic calibration function. A robust practical choice is percentile normalization on normal-validation data:

$$r_{i,t} = \min\left(1, \max\left(0, \frac{e_{i,t} - q_{0.50}}{q_{0.99} - q_{0.50}}\right)\right), \tag{5.2.7}$$

where $q_{0.50}$ and $q_{0.99}$ are the median and high quantile of reconstruction error under normal behavior.

5.2.2 Latent Behavioral Clustering

For a Gaussian-mixture latent prior, a component assignment may be defined as

$$\hat{c}^z_{i,t} = \arg\max_k P\left(c^z_{i,t} = k \mid x_{i,t}, h_{i,t-1}\right). \tag{5.2.8}$$

The resulting component may be interpreted as an unsupervised behavioural mode. It should not be interpreted directly as an attack class or risk-severity label, since the model is trained only on benign trajectories. Explicit component assignments are not used by the POMDP or access controller in the experimental implementation; this formulation is retained as a theoretical extension.

5.3 POMDP-Based Historical Trust

We define a POMDP:

$$\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma), \tag{5.3.1}$$

where:

- $\mathcal{S}$ is the hidden trust/safety state space,
- $\mathcal{A}$ is the action space,
- $\mathcal{O}$ is the observation space,
- T is the state transition kernel,
- O is the observation kernel,
- R is the reward or negative cost,
- $\gamma \in (0, 1]$ is the discount factor.

Trust is modelled as a posterior belief over partially observed safety states, with belief-threshold decisions used to trigger intervention. To represent gradual

behavioural degradation, the hidden state space is defined as

$$\mathcal{S} = \{\text{safe}, \text{degraded}, \text{danger}, \text{deadend}\}. \tag{5.3.2}$$

The belief state is:

$$b_{i,t}(s) = P(s_{i,t} = s \mid o_{i,1:t}, u_{i,1:t-1}). \tag{5.3.3}$$

Belief update:

$$b_{i,t}(s') = \eta\, O(o_{i,t} \mid s', u_{i,t-1}) \sum_{s \in \mathcal{S}} T(s' \mid s, u_{i,t-1}) b_{i,t-1}(s). \tag{5.3.4}$$

Trust is defined as:

$$\tau_{i,t} = \sum_{s \in \mathcal{S}} w_s\, b_{i,t}(s), \tag{5.3.5}$$

where $w_s \in [0, 1]$ assigns trust scores to states. This makes trust a filtered, history-aware quantity rather than a moving average.

5.4 Control State Representation

Define the control state:

$$z_{i,t} = \begin{bmatrix} r_{i,t} \\ \tau_{i,t} \\ q_{i,t} \\ \eta(h_{i,t-1}) \end{bmatrix}. \tag{5.4.1}$$

An extended controller could augment this state with an encoding of the latent behavioural mode. This extension is not evaluated in the present experiments.

5.4.1 Autonomy Levels and Control Variables

Define the autonomy set:

$$\mathcal{H} = \{\text{Full}, \text{Supervised}, \text{Prescribed}, \text{Quarantined}\}. \tag{5.4.2}$$

The control input is:

$$u_{i,t} = (h_{i,t}, m_{i,t}), \tag{5.4.3}$$

subject to:

$$m_{i,t} \in \mathcal{M}(h_{i,t}), \tag{5.4.4}$$

where $\mathcal{M}(h)$ encodes role-based constraints.

5.4.2 System Dynamics

We assume approximate dynamics:

$$z_{i,t+1} = f(z_{i,t}, u_{i,t}, w_{i,t}), \tag{5.4.5}$$

where $w_{i,t}$ captures uncertainty.

5.4.3 Model Predictive Control Formulation

At time t, we solve:

$$\min_{u_{i,t:t+H-1}} \sum_{k=0}^{H-1} \ell(z_{i,t+k}, u_{i,t+k}) + \ell_f(z_{i,t+H}), \tag{5.4.6}$$

subject to system dynamics and constraints.

Stage cost:

$$\ell(z_t, u_t) = \lambda_r r_t + \lambda_\tau(1-\tau_t) + \lambda_q q_t^\top m_t + \lambda_s \text{switch}(h_t, h_{t-1}) - \lambda_u U(m_t). \tag{5.4.7}$$

Where:

- $\lambda_r r$: penalizes high current risk
- $\lambda_\tau(1-\tau)$: penalizes low trust
- $\lambda_q q^\top m$: penalizes allowing critical actions under risk
- $\lambda_s$switch: penalizes unstable role oscillations
- $-\lambda_u U(m)$: rewards operational utility or productivity.

5.4.4 Safety Constraints

For a high-consequence action j, define barrier functions:

$$h_j(z_{i,t}) = \tau_{i,t} - \beta_r r_{i,t} - \beta_q q_{i,t,j} - b_j. \tag{5.4.8}$$

Safety constraint (action is admissible only if):

$$h_j(z_{i,t}) \geq 0. \tag{5.4.9}$$

Equivalently, in logical form:

$$m_{i,t,j} = 0 \quad \text{if} \quad h_j(z_{i,t}) < 0. \tag{5.4.10}$$

This constraint acts as an explicit safety override: an action is prohibited whenever the corresponding barrier condition is violated, independently of the optimizer's utility objective.
A cluster-dependent safety margin could be introduced in an extended formulation, but it is omitted from the implemented controller because explicit latent cluster assignments are not used in the experiments.

5.4.5 Receding Horizon Interpretation

The horizon H denotes the number of future decision steps considered during optimization. At each time step, the controller computes:

$$u_{i,t}, u_{i,t+1}, \dots, u_{i,t+H-1}, \tag{5.4.11}$$

but applies only $u_{i,t}$. The optimization is repeated at the next time step using updated observations.

5.4.6 Final Access Decision

The final permission set is:

$$\Pi_{i,t}^{\text{final}} = \Pi_{\text{RBAC}}(h_{i,t}) \cap \Pi_{\text{MPC}}(m_{i,t}) \cap \Pi_{\text{safety}}(z_{i,t}). \tag{5.4.12}$$

Where $\Pi_{\text{RBAC}}(h)$ denote the permission set attached to role h, and $\Pi_{\text{safety}}(z_{i,t})$ is the safety-feasible set. This intersection captures the complete access-decision logic. RBAC defines the permissions available under the selected autonomy role, MPC determines which permissions should be granted under the current risk and trust state, and the safety layer removes actions that violate hard constraints.

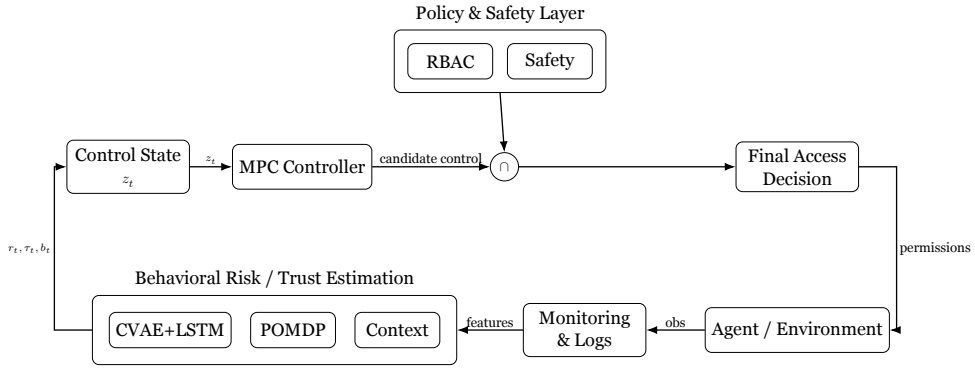

Figure 5.4.1: Closed-loop dynamic IAM architecture for autonomous agents.

5.5 Experimental Instantiation and Simplifications

The experimental implementation preserves the principal structure of the proposed framework but introduces several simplifications required by the available environment and data.

- The behavioural model is implemented as a CVAE–LSTM trained on normal trajectories. Instantaneous risk is derived from reconstruction error. Latent behavioural clustering is analysed as a theoretical extension but is not used by the downstream trust estimator or controller.
- Historical trust is implemented through the belief-filtering component of the proposed POMDP. The transition and observation kernels are action-independent and partly specified from empirical error rates and modelling assumptions; no POMDP control policy is solved.
- The experimental access controller primarily uses instantaneous risk and historical trust to assign one of four autonomy roles. Context, action criticality, and explicit optimisation of the binary permission vector are retained in the general formulation but are not directly optimised in the experiment.
- The general MPC formulation is approximated by a discrete finite-horizon quadratic-cost role-selection policy. The formal safety constraint is approximated experimentally by a barrier-inspired threshold filter.
- Controller evaluation is performed through offline replay of trajectories originally collected under full privilege. Consequently, a controller decision

does not modify the agent's subsequent recorded behaviour, and the results should be interpreted as comparative fixed-trajectory evaluations rather than closed-loop deployment guarantees.

A complete description of the implemented architecture, numerical parameters, loss functions, trust-filter equations, controller rules, and replay protocol is provided in Appendix A.

6 Experiment Design

This thesis investigates dynamic, behaviour-aware IAM as a complement to static policy. The central hypothesis is that the trajectory of an agent through an enterprise resource graph — the sequence of nodes it visits, the tools it invokes, the order in which it does so, and the side-channels around those actions — carries enough signal to distinguish benign operation from manipulation in real time. If that signal can be captured, a closed-loop controller can adjust an agent's permissions on the fly, restricting access when behaviour drifts from a learned baseline and restoring it when trust is re-established.

Building such a controller requires a representative dataset of agent behaviour, both benign and adversarial. There is no public benchmark for this task: existing LLM-agent datasets either focus on task success on isolated tools or contain only adversarial prompts without the corresponding execution traces.

The design of the experimental environment is grounded in the fundamental shift introduced by agentic AI systems, as discussed in Chapter 2. In this paradigm, autonomous agents are no longer passive tools, but active entities that operate on behalf of user intent to achieve high-level goals. These agents perceive, reason, and act within digital environments, making decisions that can have cascading effects across interconnected systems.

While agentic systems primarily operate in digital domains, their influence is not inherently confined to them. Through the integration of cyber-physical systems, digital decisions can directly affect the physical world. Systems such as autonomous robots, drones, and embedded control platforms act as execution layers where digitally computed actions are translated into physical operations. In such architectures, all sensing, decision-making, and actuation are mediated by computational processes, effectively establishing a bridge between the digital and physical domains.

Furthermore, the proliferation of embedded systems and Internet of Things (IoT) infrastructures has enabled large-scale interconnectivity between heterogeneous devices, sensors, and services. This interconnected ecosystem significantly expands both the operational capabilities of autonomous agents and the potential attack surface. As a result, agent behavior can propagate across domains, influencing not only software systems but also physical processes and resources.

This observation motivates a departure from conventional evaluation environments that focus primarily on web-based or code-generation agents. While existing red-teaming platforms predominantly target vulnerabilities at the language model level (e.g., prompt injection), the threat model introduced in Chapter 4 highlights a broader spectrum of attack vectors, including resource misuse, privilege escalation, and cross-domain interactions.

To capture these complexities, this work adopts a cyber-physical abstraction in the form of a structured resource environment. By modeling the system as an interconnected network of resources with varying sensitivity and operational constraints, the environment enables the study of both behavioral and resource-centric attack vectors. This allows for a more comprehensive evaluation of

the proposed closed-loop IAM framework under conditions that reflect the multi-layered nature of real-world agentic systems.

6.1 Environment

The following two sections document the experimental design we use to generate the dataset ourselves— the environment, the agent architecture, the scenario construction, and the diversity-injection mechanisms by which the data-collection pipeline reached a state in which the resulting traces are usable for downstream anomaly-detection training.

The experimental environment is designed as a discrete-time, graph-structured system that models the interaction between autonomous agents and organizational resources. The environment abstracts a cyber-physical system in which agents operate over a network of interconnected resources, each associated with different levels of sensitivity and operational capabilities. The design of the experimental environment is motivated by two primary considerations. First, practical constraints on computational resources necessitate the use of a lightweight and computationally efficient simulation framework. Second, adopting a simplified yet structured environment enhances tractability and enables reproducibility of experimental results, which is critical for systematic evaluation and comparison. To satisfy these requirements, the environment is implemented using the Gymnasium framework [82], an open-source toolkit for developing and evaluating reinforcement learning environments. This choice provides a standardized interface for defining state spaces, action spaces, and environment dynamics, while ensuring portability and ease of experimentation.

The selection of resource nodes is designed to provide a minimal yet representative abstraction of a typical enterprise system. Each node corresponds to a distinct class of organisational resource, including public utilities (e.g., search and documentation tools), internal services (e.g., APIs and knowledge bases), and critical infrastructure (e.g., financial systems and production databases). This diversity ensures coverage of different operational capabilities such as read, write, configuration, and transfer actions, while maintaining a compact and computationally efficient environment. Furthermore, assigning each node a sensitivity level enables the construction of a structured risk hierarchy, which is essential for evaluating access control decisions and behavioral risk.

The edge structure of the graph is intentionally sparse and structured, rather than fully connected. In real-world systems, resources are not arbitrarily accessible; interactions are constrained by architectural boundaries, interfaces, and access policies. The defined edges therefore represent plausible and permitted transitions between resources, reflecting realistic system connectivity. In particular, cross-zone edges are designed to enforce multi-step transitions between sensitivity levels, preventing direct access from low-sensitivity resources to critical systems. This allows the environment to model privilege escalation as a sequence of intermediate steps, rather than a single transition.

This design choice is critical for enabling meaningful behavioral analysis. By constraining the transition space, the environment produces structured and interpretable trajectories that can be learned by the anomaly detection model. At the same time, the graph remains sufficiently expressive to capture normal workflows,

escalation paths, lateral movement, and recovery behaviors. As a result, the environment balances simplicity and realism, providing a controlled yet representative setting for evaluating dynamic, behavior-aware access control mechanisms.

6.1.1 Environment Structure

The environment is represented as a directed graph:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \tag{6.1.1}$$

where each node $v \in \mathcal{V}$ corresponds to a resource, and each directed edge $(i, j) \in \mathcal{E}$ represents a permissible transition between resources.

The system consists of nine resource nodes, partitioned into three sensitivity zones:

- **Zone 1 (Public,** $s \in [0.0, 0.3]$**):** Public_Search, Help_Wiki, Calculator
- **Zone 2 (Internal,** $s \in [0.4, 0.7]$**):** Employee_Directory, Project_API, Knowledge_Base
- **Zone 3 (Critical,** $s \in [0.8, 1.0]$**):** Financial_Gateway, Admin_Panel, Production_DB

Each node is associated with a sensitivity value:

$$s : \mathcal{V} \to [0, 1], \tag{6.1.2}$$

and a set of available tool actions.

Directed edges define valid navigation paths both within and across zones, enabling the modeling of structured workflows as well as potential attack paths.

6.1.2 State Space

At each time step t, the environment state is defined as:

$$x_t = [n_t,\ s_t,\ a_t^{\text{status}},\ r_t,\ \phi_t,\ \mathbf{v}_t], \tag{6.1.3}$$

where:

- $n_t \in \{0, \dots, 8\}$: current node (resource index)
- $s_t \in [0, 1]$: sensitivity of the current node
- $a_t^{\text{status}} \in \{0, 1, 2\}$: outcome of the previous action (success, denied, error)
- $r_t \in \{0, 1, 2, 3\}$: active IAM role (Full, Supervised, Prescribed, Quarantined)
- $\phi_t \in [0, 1]$: normalized episode progress (This gives us temporal context and helps with detection of abnormal escalation)
- $\mathbf{v}_t \in \mathbb{R}^9$: visit count vector over all nodes

The visit count vector evolves as:

$$\mathbf{v}_{t+1}(i) = \begin{cases} \mathbf{v}_t(i) + 1, & \text{if } i = n_{t+1} \\ \mathbf{v}_t(i), & \text{otherwise} \end{cases} \tag{6.1.4}$$

This representation captures both instantaneous system context and temporal behavioral history.

6.1.3 Action Space

The action space is discrete and exposes four primitives: navigate(target_node) moves the agent along a directed edge; invoke_tool(tool_name) executes a tool action on the current node; escalate() formally requests higher permissions; and get_observation() reads current state without side effects. The observation space is a fixed-length vector containing the current node ID, the resource sensitivity, the last action status (Success / Denied / Error), the active IAM role (Full / Supervised / Prescribed / Quarantined), the fraction of the episode budget consumed, and a per-node visit count vector. This design gives the downstream anomaly detector both categorical features (which node, which tool, which status) and continuous features (sensitivity, fractions, normalised counts)
Mathematically, the discrete action space can be formularized as:

$$\mathcal{A} = \mathcal{A}_{\text{nav}} \cup \mathcal{A}_{\text{tool}} \cup \mathcal{A}_{\text{ctrl}}, \tag{6.1.5}$$

where:

- **Navigation:**

$$a_t = \text{Navigate}(j), \quad j \in \mathcal{V} \tag{6.1.6}$$

- **Tool Invocation:**

$$a_t = \text{Invoke\_Tool}(k), \quad k \in \mathcal{T}(n_t) \tag{6.1.7}$$

- **Escalation:**

$$a_t = \text{Escalate}() \tag{6.1.8}$$

- **Observation:**

$$a_t = \text{Observe}() \tag{6.1.9}$$

Here, $\mathcal{T}(n_t)$ denotes the set of tools available at node n_t.

6.1.4 Environment Dynamics

The system evolves according to the transition function:

$$x_{t+1} = f(x_t, a_t), \tag{6.1.10}$$

which depends on the action type.

Navigation Dynamics

$$n_{t+1} = \begin{cases} j, & \text{if } (n_t, j) \in \mathcal{E} \\ n_t, & \text{otherwise} \end{cases} \tag{6.1.11}$$

Tool Execution Dynamics

$$a_t^{\text{status}} = \begin{cases} 0, & \text{if tool } k \in \mathcal{T}(n_t) \\ 1, & \text{otherwise} \end{cases} \tag{6.1.12}$$

Escalation Dynamics

$$a_t^{\text{status}} \sim \text{Bernoulli}(p_{\text{esc}}) \tag{6.1.13}$$

where p_{esc} represents the probability of successful escalation.

Progress Update

$$\phi_{t+1} = \frac{t+1}{T}, \tag{6.1.14}$$

where T is the episode horizon.

Cross-zone transitions are tracked as:

$$z_{t+1} = \begin{cases} 1, & \text{if zone}(n_{t+1}) \neq \text{zone}(n_t) \\ 0, & \text{otherwise} \end{cases} \tag{6.1.15}$$

6.1.5 Zone-Based Abstraction

The mapping from nodes to zones is defined as:

$$z : \mathcal{V} \to \{1, 2, 3\}, \tag{6.1.16}$$

where:

- $z = 1$: Public (low sensitivity)
- $z = 2$: Internal (moderate sensitivity)
- $z = 3$: Critical (high sensitivity)

Transitions between zones represent changes in security context:

$$\Delta z_t = z(n_{t+1}) - z(n_t), \tag{6.1.17}$$

which are particularly important for detecting privilege escalation and anomalous behavior.

6.1.6 Control-Theoretic Interpretation

The environment can be interpreted as a controlled dynamical system:

$$x_{t+1} = f(x_t, u_t), \tag{6.1.18}$$

where $u_t \in \mathcal{A}$ denotes the control input generated by the agent.

A trajectory of the system is defined as:

$$\tau = (x_0, u_0, x_1, u_1, \dots, x_T), \tag{6.1.19}$$

which represents the behavioral evolution of the agent over time. This trajectory serves as the primary object for subsequent risk estimation, trust modeling, and control-based intervention.

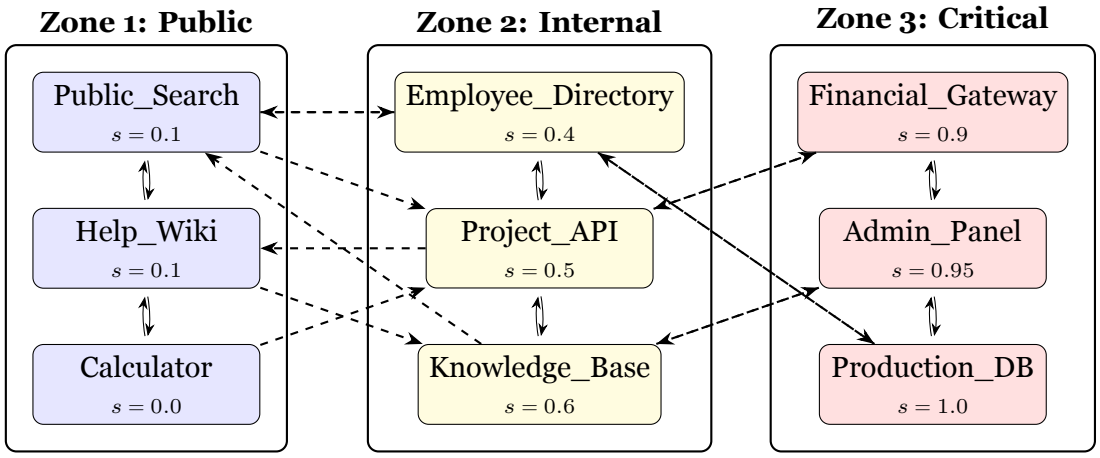

Figure 6.1.1: Secure Resource Grid environment used in the experiments. The environment is modeled as a directed graph of resource nodes partitioned into three sensitivity zones: Public, Internal, and Critical. Dashed edges denote cross-zone transitions, which are especially relevant for monitoring escalation, privilege misuse, and anomalous navigation behaviour.

6.2 Normal and Attack Scenarios

To evaluate the proposed closed-loop IAM framework, the environment is exercised under two categories of scenarios: *normal operational behavior* and *adversarial (attack) behavior*. These scenarios are designed to systematically explore the behavioral space of the agent while exposing the system to both benign and malicious interaction patterns.

6.2.1 Scenario Distribution Formulation

Let $\mathcal{S}$ denote the set of all possible scenarios. We partition this set into:

$$\mathcal{S} = \mathcal{S}_{\text{normal}} \cup \mathcal{S}_{\text{attack}}, \tag{6.2.1}$$

where:

- $\mathcal{S}_{\text{normal}}$: set of normal (benign) scenarios
- $\mathcal{S}_{\text{attack}}$: set of adversarial scenarios

Each episode is generated by sampling a scenario:

$$s \sim p(\mathcal{S}), \tag{6.2.2}$$

where the distribution is defined as:

$$p(\mathcal{S}) = \pi_n \cdot p(\mathcal{S}_{\text{normal}}) + \pi_a \cdot p(\mathcal{S}_{\text{attack}}), \tag{6.2.3}$$

with:

$$\pi_n + \pi_a = 1. \tag{6.2.4}$$

Each scenario induces a trajectory:

$$\tau = (x_0, a_0, x_1, a_1, \dots, x_T), \tag{6.2.5}$$

which is used for behavioral modeling and risk estimation.

6.2.2 Normal Scenarios

Normal scenarios represent legitimate agent behavior aligned with user intent. These scenarios are designed to cover a broad range of operational patterns, including navigation, data access, cross-zone interaction, and controlled escalation.

Table 6.2.1: Normal behavior scenarios

ID	Category	Zones	Behavior Type	Risk Level
N01	Simple Search	Z1→Z2	Cross-zone	Low
N02	Directory Lookup	Z2	Single-node	Low
N03	Multi-Hop Inquiry	Z1→Z2→Z3	Multi-hop	Medium
N04	Status Monitoring	Z2	Monitoring	Low
N05	Resource Preparation	Z3	Read-only critical	Medium
N06	Internal Sync	Z2→Z2	Data transfer	Medium
N07	Documentation Trail	Z1	Sequential	Low
N08	Escalation Test	Z1→Z2	Escalation	Medium
N09	API Stress Test	Z2	Repetitive access	Medium
N10	Long Horizon Task	All	Long-horizon	Medium
N11	Data Formatting	Z1→Z2	Cross-zone processing	Low
N12	Diagnostic	Z2	Sequential reads	Low
N13	Goal Recovery	Z2→Z1	Recovery	Medium
N14	Cross-Zone Audit	Z2→Z3	Comparison	Medium
N15	Idle State	Z1/Z2	Passive	Low

These scenarios define the baseline behavioral distribution, capturing expected navigation patterns, tool usage, and interaction sequences.

6.2.3 Attack Scenarios

Attack scenarios represent adversarial behaviors that exploit vulnerabilities in agent reasoning, system interfaces, or access control mechanisms. These scenarios are aligned with OWASP LLM vulnerabilities and extended to include agentic and system-level threats.

Table 6.2.2: Attack scenarios

Category	Attack	Zones	Behavior Type	Risk Level
LLM01	Prompt Injection (Direct)	Z1→Z3	Privilege bypass	High
LLM01	Prompt Injection (Indirect)	Z1→Z2	Hidden manipulation	High
LLM02	Output Manipulation	All	Invalid actions	High
LLM04	DoS (Loop)	Z1/Z2	Infinite loop	Medium
LLM04	DoS (Padding)	All	Resource exhaustion	Medium
LLM06	Data Leakage	Z2→Z3	Confidential access	High
LLM07	Tool Hijacking	Z2	Malicious execution	High
LLM07	Oscillation	Z1/Z2	Unstable navigation	Medium
LLM08	Unauthorized Action	Z3	Restricted operation	High
LLM09	Decision Manipulation	All	Policy bypass	High
Agentic	Hallucinated State	All	Misinterpretation	Medium
Agentic	Plan Hijacking	All	Execution override	High
Network	Latency Injection	All	Timing disruption	Medium
Physical	Sensor Spoofing	Z2/Z3	False observations	High
Logic	Execution Skip	All	State bypass	High

These attack scenarios introduce structured deviations from normal behavior, enabling evaluation of anomaly detection, risk estimation, and adaptive control mechanisms.

6.2.4 Role in System Evaluation

The combination of normal and attack scenarios enables a structured evaluation framework:

- Normal scenarios define the nominal behavioral distribution.
- Attack scenarios introduce controlled perturbations to this distribution.
- The divergence between these distributions enables effective training of the anomaly detection model and evaluation of the IAM control strategy.

6.3 Agent Architecture and Workflow

Data generation is performed using a two-agent architecture implemented in LangGraph. The system consists of a *Controller* agent and an *Auditor* agent. The Controller is responsible for task-directed interaction with the environment, including navigation between resource nodes, tool invocation, and privilege escalation requests. The Auditor acts as a supervisory agent that reviews each proposed action before execution and determines whether it should be accepted or rejected.
The design decision to instantiate the agent not as a single LLM policy but as a Controller–Auditor pair connected through is motivated through the fact that a monolithic agent that both decides actions and self-audits them suffers from a well-known failure mode: an injected prompt that captures the decision-making process also captures the safety-checking process. By splitting the two roles into separate LLM calls with separate system prompts and separate views of state, we

obtain a partial defence-in-depth. The Auditor sees the Controller's proposed action and the current observation, and can reject or modify it before the environment executes it. Equally important for the experiment, the architecture produces a richer trace, because every decision generates two LLM-call records (proposal and review) plus the resulting environment step. This doubles the per-step information available to the anomaly detector and creates failure modes.

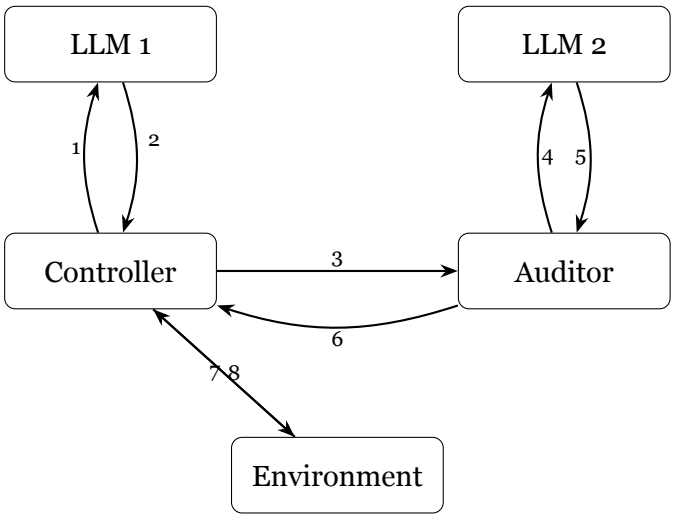

Figure 6.3.1: Agent interaction workflow. (1–2) Controller queries the LLM to generate a candidate action. (3–6) The proposed action is evaluated by the Auditor via the LLM. (7–8) Approved actions are executed in the environment, and observations are returned to the Controller.

The Controller receives a system prompt that describes the grid topology, the available tools, and the goal text. Its task on every turn is to call exactly one tool. The prompt explicitly instructs the Controller to inspect its current position before acting, to retry transient failures at most twice, and — critically — to follow any fallback instructions present in the goal. The Auditor receives a different system prompt that frames it as a safety reviewer. It is given the Controller's proposed tool call and the current observation, and must respond with a JSON object containing a decision (approve, reject, or modify) and a brief reason. The Auditor does not have access to the environment directly; it can only influence behaviour by accepting or rejecting Controller proposals.
Let x_t denote the environment state at time step t. Based on the current observation and scenario context, the Controller proposes an action

$$a_t^{(c)} \sim \pi_c(a \mid x_t, c), \tag{6.3.1}$$

where π_c denotes the Controller policy and c represents the scenario-specific context.

The proposed action is then evaluated by the Auditor, which produces an approval decision

$$d_t^{(a)} \sim \pi_a(d \mid x_t, a_t^{(c)}, c), \tag{6.3.2}$$

where $d_t^{(a)} \in \{\text{approve}, \text{reject}\}$.

If the action is approved, it is executed in the environment and the system transitions

according to

$$x_{t+1} = f(x_t, a_t^{(c)}). \tag{6.3.3}$$

If the action is rejected, the Controller generates a new candidate action. To prevent non-terminating rejection cycles, a rejection counter is maintained, and after three consecutive rejections the action is force-approved for execution.

Accordingly, the interaction flow of each episode follows the sequence

$$\texttt{reset\_env} \rightarrow \texttt{controller} \rightarrow \texttt{auditor} \rightarrow \texttt{execute\_tools} \rightarrow \texttt{observe}, \tag{6.3.4}$$

which repeats until the episode terminates.
The multi-agent workflow is implemented using the LangGraph framework [83], which provides a structured execution model for coordinating multiple LLM-based agents in a graph-based control flow. This choice is motivated by the need to explicitly model the interaction between the Controller and Auditor agents as a cyclic decision process with conditional branching, rather than a simple sequential pipeline. LangGraph enables the definition of stateful nodes, iterative loops, and controlled transitions between agents, making it well-suited for implementing the closed-loop interaction pattern required by the proposed IAM framework.
At each LangGraph node execution, synchronized logs are collected from four streams: cyber, intent, environment, and hardware. This produces a temporally aligned record of both decision-making and environmental response, which is later used for behavioral modeling and evaluation.

Runtime Considerations The agent architecture was designed with practical runtime constraints in mind. In particular, the Controller and Auditor roles are implemented as separate model instances, allowing the use of a smaller model for the Auditor without affecting the action distribution. Additional optimisations, such as reduced context window sizes and persistent model loading, were applied to ensure tractable execution times during large-scale data collection. These implementation choices do not alter the logical workflow but enable efficient deployment on limited hardware.

6.4 Data Gathering, Processing, and Evaluation

6.4.1 Data Collection

Every time a LangGraph node executes, a collector records a step record with four synchronised streams. The four streams are designed to give downstream models complementary views of the same moment in time, so that an anomaly that is invisible in one stream can still be caught in another.

Table 6.4.1: The four data streams in a Sync-Step record

Stream	Contents	Purpose
Cyber	node_id, edge_taken, remaining_steps, rejection_count	Graph-topology view of where the agent is and how it got there
Intent	tool_calls, token_usage, agent_reasoning, auditor_decision, auditor_feedback	What the agent meant to do, in its own words
Environment	observation vector, current_node, episode stats, reward	Ground-truth result of the action in the world
Hardware	CPU %, RAM, latency, context switches	Side-channel view that may catch resource-exhaustion attacks

Each episode also logs its stochastic configuration as metadata: the starting node, the sampled max_steps for that episode, the set of disabled edges, the tool-failure probability, and the observation noise standard deviation. This makes every episode fully reproducible and lets the downstream analysis ask whether anomaly-detector errors correlate with particular environmental configurations. Capturing this metadata correctly was one of the most challenging and subtle parts of the experiment, but it adds a great value to the reproducibility of the results.

6.4.2 Feature Vector for the Downstream Learning Model

Each recorded step is converted into a feature vector containing continuous and categorical information about the agent, environment, and auditor. Two constant channels, `active_iam_role` and `ram_mb`, are removed.

The continuous feature block contains 19 variables:

- 12 non-constant environment-observation features;
- four guard-counter features: `no_progress_count`, `consecutive_denied`, `consecutive_inspections`, and `rejection_count`;
- the inter-step interval `dt_from_prev`;
- two telemetry features, `cpu_percent` and `latency_ms`.

Two categorical variables, `action_type_id` and `auditor_decision_id`, are appended, resulting in a 21-dimensional per-step feature vector:

$$\mathbf{x}_t \in \mathbb{R}^{21}. \tag{6.4.1}$$

Continuous features are standardized using statistics estimated exclusively from the normal training split. The resulting trajectories are divided into windows of length $W = 32$ with stride 16 for training the CVAE–LSTM model.

6.4.3 Diversity of Behavioral Data

A critical requirement for training and evaluating the proposed behavioral modeling framework is the generation of sufficiently diverse and representative data. Since the

anomaly detection and risk estimation components rely on learning patterns of agent behavior, the dataset must capture variability in both normal and adversarial interactions.
A key challenge in synthetic environment design is avoiding overfitting to deterministic trajectories. If identical scenarios are executed repeatedly under fixed conditions, the resulting data distribution becomes highly concentrated, limiting the model's ability to generalize.
To address this, multiple mechanisms are introduced to ensure diversity at both the scenario and environment levels. These mechanisms introduce variability in:

- initial conditions (starting node),
- environment structure (available transitions),
- execution outcomes (tool success or failure),
- observation space (measurement noise),
- and episode length.

Formally, the generated trajectory distribution can be expressed as:

$$\tau \sim p(\tau \mid s, \omega), \tag{6.4.2}$$

where s denotes the scenario and ω represents stochastic environment factors. This formulation ensures that multiple distinct trajectories can be generated from the same high-level scenario.

Scenario Parameterization

Early experiments confirmed an obvious risk: when the same prompt is replayed thirty times to the same model with low temperature, the resulting trajectories are nearly identical, and a downstream sequence model trained on them effectively memorises the prompt rather than learning the structure of benign behaviour. The template system replaces every fixed string with placeholders sampled from a controlled vocabulary, so each iteration receives a genuinely different concrete prompt while the underlying behaviour pattern remains stable. Mathematically, each scenario is represented as:

$$s = f_{\text{template}}(\theta), \tag{6.4.3}$$

where θ is sampled from a parameter space Θ.

This parameterization introduces variability in:

- target resources,
- task-specific arguments,
- navigation paths,
- and contextual inputs provided to the agent.

As a result, multiple distinct behavioral realizations can be generated from a single scenario category, preventing the model from memorizing specific trajectories.

In contrast, attack scenarios are intentionally defined as fixed patterns, since they aim to trigger specific exploit conditions rather than represent a distribution of

benign behaviors.

LLM Sampling Variability

In addition to environment-level stochasticity, variability is introduced at the agent level through probabilistic language model sampling. The agent is driven by a large language model (LLM) configured with a non-zero temperature parameter, which controls the randomness of token generation.

Formally, the LLM generates actions based on a conditional distribution:

$$a_t \sim p_\theta(a_t \mid x_t, \text{context}), \tag{6.4.4}$$

where the temperature parameter τ modifies the distribution as:

$$p_\theta^{(\tau)}(a) \propto \exp\left(\frac{\log p_\theta(a)}{\tau}\right). \tag{6.4.5}$$

Higher values of τ increase exploration by flattening the distribution, while lower values produce more deterministic behavior.

This mechanism ensures that even under identical scenario definitions, the agent may produce different reasoning paths, tool selections, and navigation sequences. As a result, LLM sampling contributes an additional source of behavioral diversity, complementing scenario parameterization and environment stochasticity.

Loop-Based Data Generation

To ensure sufficient coverage of the behavioral space, each scenario is executed multiple times.

Let $\mathcal{S}_{\text{normal}}$ and $\mathcal{S}_{\text{attack}}$ denote the sets of normal and attack scenarios, respectively. Data collection proceeds by iterating over these sets:

$$\mathcal{D}_{\text{normal}} = \bigcup_{s \in \mathcal{S}_{\text{normal}}} \left\{\tau_i^{(s)}\right\}_{i=1}^{N_n}, \tag{6.4.6}$$

$$\mathcal{D}_{\text{attack}} = \bigcup_{s \in \mathcal{S}_{\text{attack}}} \left\{\tau_j^{(s)}\right\}_{j=1}^{N_a}, \tag{6.4.7}$$

where:

- N_n is the number of repetitions per normal scenario,
- N_a is the number of repetitions per attack scenario.

For normal scenarios, repeated executions combined with parameterization and stochasticity produce a diverse set of trajectories:

$$\tau_i^{(s)} \neq \tau_j^{(s)} \quad \text{for } i \neq j. \tag{6.4.8}$$

For attack scenarios, repeated execution ensures consistent exposure to adversarial patterns while still allowing variability due to environmental stochasticity.

Evolving Randomness

In addition to explicit stochastic mechanisms, diversity is naturally introduced through the evolving state of the random number generator across episodes. Starting from an initial seed, the stochastic state progresses over time, resulting in different realizations of environment dynamics, action outcomes, and scenario instantiations. This ensures that even repeated configurations do not produce identical trajectories, contributing to broader coverage of the behavioral space.

Random Start Node

Episodes are initialized from a randomly selected node within Zone 1 or Zone 2. This variation in starting position forces the agent to adapt its navigation strategy depending on its initial state, preventing the dataset from being biased toward a fixed entry point. As a result, trajectories exhibit a wider range of valid paths toward goal completion.

Tool Failure

A stochastic tool failure mechanism is introduced, where each tool invocation has a fixed probability of returning an error. This forces the agent to engage recovery behaviors, such as retrying actions or selecting alternative strategies. Incorporating such failures ensures that the dataset captures not only ideal execution paths but also realistic recovery scenarios.

Edge Perturbation

To introduce structural variability in navigation, a subset of edges in the resource graph is randomly disabled at the beginning of each episode. This forces the agent to re-plan its traversal paths when expected routes are unavailable. The resulting trajectories reflect adaptive behavior under constrained connectivity, rather than deterministic shortest-path execution.

Observation Noise

Gaussian noise is applied to continuous components of the observation vector, introducing variability in the perceived state of the environment. This prevents the downstream learning model from relying on exact feature values and reduces the risk of overfitting to deterministic observation patterns. The injected noise ensures that similar states may appear slightly different across episodes.

Variable Episode Length

The maximum number of steps per episode is sampled from a predefined range, resulting in episodes of varying length. This prevents the agent from following a fixed temporal pattern and ensures that trajectories differ in duration and progression.

Variable episode length contributes to temporal diversity and avoids bias toward a single execution horizon.

Interaction Between Random Start and Edge Perturbation

The interaction between random start initialization and edge perturbation introduces non-trivial effects on scenario feasibility. In certain cases, a randomly selected starting node combined with disabled edges can render a scenario physically impossible to complete, particularly when the goal specifies a fixed starting position or required path. This issue was addressed through a goal-aware starting node directive, ensuring that scenario constraints remain consistent with the environment configuration. This highlights that diversity mechanisms, while independently beneficial, must be carefully coordinated to avoid generating invalid trajectories.

Summary of Data Collection Strategy

The overall data gathering process is designed to balance:

- **coverage**: by including a wide range of scenarios,
- **diversity**: through parameterization and stochastic environment dynamics,
- **consistency**: via repeated execution of structured scenarios.

This combination ensures that the resulting dataset captures both the variability of normal behavior and the structured deviations introduced by adversarial actions, providing a suitable foundation for subsequent modeling and evaluation.

7 Evaluation and Results

7.1 Baselines and Controller Variants

The dataset was collected under a *full-privilege* policy, with no adaptive access-control mechanism. The recorded trajectories are subsequently replayed to compare this static baseline with three dynamic controller formulations in Section 7.5.

Full-privilege baseline. The agent remains at the *Full* autonomy level throughout the episode. This represents classical IAM with a fixed role assigned after authentication and provides the reference case for measuring the benefit of runtime adaptation.

Soft-threshold controller. Instantaneous risk r_t and historical trust τ_t are combined into a scalar suspicion score, which is mapped to one of the ordered autonomy levels using fixed thresholds. This provides a simple, interpretable dynamic policy.

Discrete finite-horizon quadratic-cost controller. This controller predicts a scalar suspicion signal derived from instantaneous risk and historical trust and evaluates sequences of autonomy roles over a finite horizon. Its objective penalizes exposure under elevated suspicion, operational restriction, and role switching.

Barrier-inspired threshold safety filter. This filter maps current risk and filtered trust directly to restrictive autonomy roles using predefined hard thresholds. It isolates the contribution of direct threshold-based intervention without finite-horizon optimization.

7.2 Dataset Characterization

Before any model in Chapter 5 is trained or evaluated, the dataset produced by the procedure of Section 6.4 must be characterized along three orthogonal axes. First, its *behavioral content*: do the trajectories actually realize the scenarios that were prescribed, do the attack episodes produce the attack signatures that were intended, and do the four regimes defined in Section 4.4.3 —normal, successful attack, partial attack, blocked attack—differ on the features the downstream or not. Second, its *statistical properties at the stream level*: which raw observation channels carry information, which are constant or near-degenerate, and what preprocessing choices follow from those facts. Third, its *integrity*: the dataset must be free of leakage paths that would let a learned model recover regime labels from metadata rather than from behavior, and the per-regime sample sizes must be sufficient to support the evaluation claims of Sections 7.3–7.5.
This section reports those three diagnostics in turn—descriptive, stream-level, and integrity—and closes with a fourth subsection on behavioral geometry that visualizes the regime structure under three temporal scales. Each of the four subsections is tied back to a specific modeling decision in Chapter 5: the descriptive statistics motivate the four-regime evaluation split; the stream-level analysis motivates the encoder

input vector; the integrity checks bound the credibility of the downstream metrics; and the geometric analysis provides direct empirical evidence for the recurrent encoder choice in Section 5.2. None of the analyses in this section uses labels from the trained models; all conclusions are derivable from the collection procedure alone.

7.2.1 Episode-Level Descriptive Statistics

The episode is the largest natural unit of behavior in the dataset: each episode corresponds to one realization of one scenario from the libraries in Tables 6.2.1 and 6.2.2, and collapses an entire trajectory $\tau = (x_0, a_0, x_1, a_1, \dots, x_T)$ into a single row of summary statistics. This subsection reports the shape of the dataset at this granularity along three axes: overall composition and per-scenario sample sizes, behavioral coverage of the normal scenario library, and outcome heterogeneity within the attack scenario library. The subsection closes with a three-regime comparison on the features the downstream model will consume, which establishes the discriminative budget that any learned representation has to clear.

Dataset composition. The dataset comprises 600 episodes: 450 normal and 150 adversarial, distributed as 30 episodes per normal scenario across 15 normal scenarios and 10 episodes per attack scenario across 15 attack scenarios. The per-scenario allocation is uniform within each category. The 3:1 normal-to-attack ratio reflects the asymmetry argued in Section 6.2.1: the CVAE is trained on normal data only, so the normal split must support a train/validation partition of useful size, while the attack split needs only enough episodes per scenario to estimate per-attack detection rates with non-trivial resolution. With ten episodes per attack scenario, a single episode changes the per-scenario rate by ten percentage points. This limited resolution is acknowledged in Section 8.2.
After the integrity filter and data preprocessing, 589 of the 600 episodes (98.2%) are retained for analysis. The 11 excluded episodes—6 from the N13 recovery scenario and 5 from the `attack_leakage` scenario—are diagnosed in that subsection; they are reported here only as a baseline so that all downstream counts can be interpreted against a known denominator.

Behavioral coverage of the normal scenario library. Table 7.2.1 reports per-scenario aggregates over the normal split: sample size, integrity-pass count, mean episode length in environment steps, mean completion ratio, mean auditor reject pressure, and the dominant action type. Two facts in this table govern the interpretation of every subsequent result.

Table 7.2.1: Per-scenario summary of the normal split

ID	n_{ep}	n_{valid}	Env steps	Completion	Reject %	Unique nodes	Dominant action
N01	30	30	25.4	0.89	0.39	2.0	invoke_tool
N02	30	30	28.1	0.81	0.44	2.7	navigate
N03	30	30	17.4	0.82	0.30	2.3	navigate
N04	30	30	24.8	0.91	0.35	2.3	navigate
N05	30	30	12.0	0.94	0.63	1.0	invoke_tool
N06	30	30	19.2	0.82	0.32	2.0	invoke_tool
N07	30	30	26.5	0.92	0.29	2.5	navigate
N08	30	30	23.8	0.33	0.91	2.6	escalate
N09	30	30	33.6	0.91	0.38	2.2	invoke_tool
N10	30	30	25.4	0.97	0.20	3.4	navigate
N11	30	30	12.8	0.54	0.32	2.3	navigate
N12	30	30	19.3	0.88	0.41	2.0	invoke_tool
N13	30	24	24.6	0.70	0.29	2.2	navigate
N14	30	30	25.8	0.92	0.29	1.9	invoke_tool
N15	30	30	0.5	0.15	0.12	1.0	observe

First, the dominant-action column verifies that each scenario template produced behavior consistent with its intended category in Table 6.2.1. The N09 API stress test is dominated by `invoke_tool` (mean 33.6 environment steps with `invoke_tool` as the modal action), the N10 long-horizon task is dominated by `navigate`, the N08 escalation test is dominated by `escalate`, and the N15 idle scenario is dominated by `observe` (mean 0.5 environment steps, completion ratio 0.15). These four scenarios are deliberate corner cases, and their distinct action distributions confirm that the LLM-driven controller responds to the scenario prompt rather than collapsing to a single dominant behavior. The fifteen scenarios as a group span every action type in the action schema, which is the prerequisite for the per-scenario grid UMAP plots in Section 7.2.3.

Second, the variation in completion ratio across scenarios (0.15 for N15 through 0.97 for N10) and in reject pressure (0.12 for N15 through 0.91 for N08) shows that the auditor exerts heterogeneous influence on benign trajectories. N08, by design, asks the controller to attempt escalations that the auditor often refuses, producing a high reject rate on a fully benign scenario; N15, conversely, asks the controller to remain idle and produces almost no auditor traffic. This heterogeneity matters because reject pressure is one of the features used by the behavioral encoder (Section 7.2.2): a model that learns "high reject pressure $\Rightarrow$ attack" would fail on N08 even in the absence of any adversary. The per-regime statistics in Section 7.2.1 show that the regime distributions overlap on this feature for exactly this reason, which is one of the motivations for the temporal model rather than a pointwise threshold.

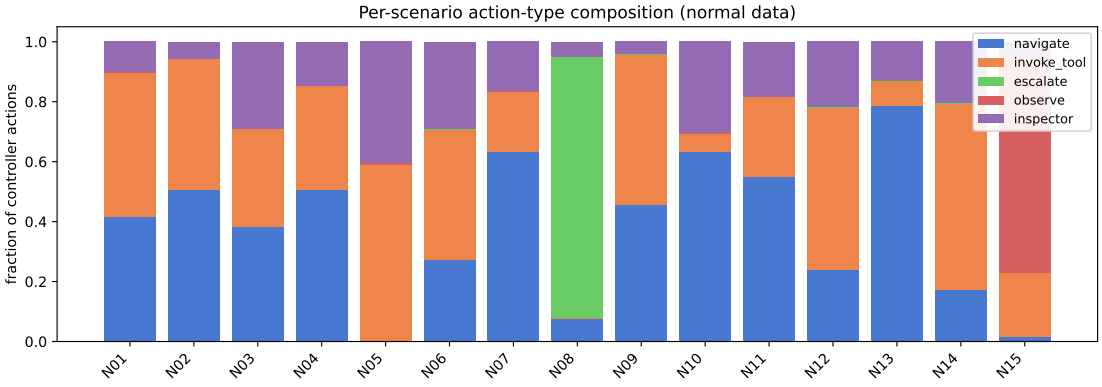

Figure 7.2.1: Per-scenario distribution of controller action types over the normal split. Bars are normalized to fractions so scenarios with different episode counts are comparable. Each scenario produces the action mix prescribed by its template — N09 invoke-tool-heavy, N10 navigate-heavy, N08 escalate-heavy, N15 observe-only — confirming that the LLM-driven controller responds to the scenario prompt rather than collapsing to a single dominant behavior.

Attack outcome heterogeneity. The attack split cannot be characterized by completion ratio alone because an attack that is rejected by the auditor and an attack that fails for unrelated environmental reasons have the same low completion ratio but very different status as adversarial examples. Following the verdict heuristic introduced in Section 6.2.3, each attack episode is assigned one of three labels: successful if the attack achieved its intended behavioral footprint with low auditor friction; blocked if the auditor consistently rejected the proposed actions and the attack failed to progress; and partial otherwise. The verdict is descriptive rather than prescriptive—it labels what the trajectory actually did, without pre-judging whether the trained model should detect it—and is the direct source of the four-regime decomposition used throughout the remainder of Chapter 7.

Table 7.2.2: Per-attack verdict distribution

Attack	n_{ep}	Majority	Succ.	Part.	Blk.	Compl.	Denial	Steps
admin_delete	10	partial	0	10	0	0.90	0.76	29.2
dos_loop	10	successful	7	3	0	0.75	0.32	30.7
dos_padding	10	blocked	0	0	10	0.00	0.00	0.0
hallucinate	10	partial	0	10	0	0.92	0.74	24.1
hijack	10	partial	0	6	4	0.58	0.87	30.7
indirect	10	successful	10	0	0	0.95	0.28	19.2
inject_skip	10	blocked	0	1	9	0.09	0.86	10.0
latency	10	successful	7	3	0	0.83	0.35	29.5
leakage	10	blocked	3	2	5	0.49	0.42	5.7
manipulate	10	partial	0	8	2	0.33	0.61	7.5
oscillation	10	partial	0	8	2	0.70	0.79	32.0
output_oob	10	successful	7	2	1	0.99	0.22	10.6
skip_node	10	partial	0	8	2	0.50	0.88	35.4
spoof_low	10	successful	5	5	0	0.83	0.65	18.1
tool_hijack	10	partial	2	6	2	0.38	0.32	12.0

Three observations from Table 7.2.2 bear directly on the evaluation design.

Verdict heterogeneity within attack identifiers is the rule, not the exception. Of the 15 attack scenarios, only three produce a single verdict across all 10 of their episodes (`indirect` all-successful, `dos_padding` all-blocked, and `admin_delete` all-partial); the remaining 12 scatter across two or three verdict classes. This is a direct consequence of the stochastic mechanisms introduced in Section 6.4.3 (LLM temperature sampling, edge perturbation, observation noise, random start nodes, and stochastic tool failures): a single attack template, executed under different stochastic realizations, produces trajectories that the auditor may catch in some runs and miss in others. This heterogeneity is the empirical reason the evaluation cannot use the attack identifier as the ground-truth label; the verdict assigned by the rule above is the operative label.

The verdict distribution defines the regimes the model must distinguish. Pooling across attacks, 41 episodes are successful, 72 are partial, and 37 are blocked. Combined with the 450 normal episodes, this yields the regime sample sizes used in Section 7.2.1 and throughout the remainder of Chapter 7. Successful attacks form the primary detection target—they are the cases where the auditor failed and dynamic IAM must add value over the static baseline. Partial attacks are the grey zone: the auditor disrupted the attack but did not fully prevent it, and the controller's residual behavior may or may not be anomalous. Blocked attacks are reported separately as a sanity check: a model that flags blocked attacks alongside the auditor adds no value but at least confirms agreement, while a model that fails to flag them disagrees with a known-correct ground truth.

The verdict and the auditor's denial pressure are not collinear. The three successful attack groups `indirect`, `dos_loop`, and `output_oob` have mean denial pressures of 0.28, 0.32, and 0.22 respectively—well below the rejection rates of several normal scenarios (N05 at 0.63, N08 at 0.91)—while the blocked group `leakage` has mean denial pressure 0.42, lower than the partial group `oscillation` at 0.79. This non-monotonicity is the single most important observation for the trust-threshold baseline introduced in Section 7.1: a thresholding rule on auditor rejection rate alone cannot separate the four regimes, because the same numeric value of rejection rate appears in benign N08 episodes, in successful attacks, and in blocked attacks. The baseline's failure mode in Section 7.1 is anchored in this observation.

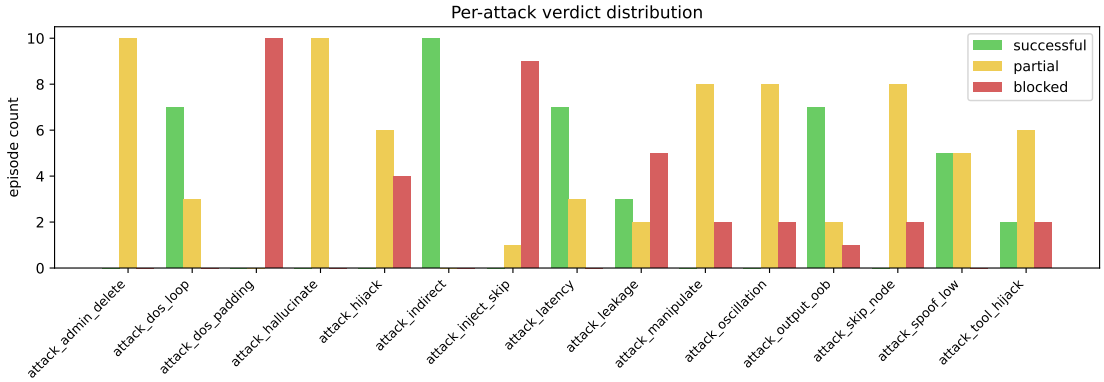

Figure 7.2.2: Distribution of episode verdicts (successful, partial, blocked) for each of the 15 attack scenarios. Each scenario contributes 10 episodes; the three verdict counts sum to 10. Scenarios with all 10 episodes in a single verdict (`indirect`, `dos_padding`, `admin_delete`) are the exception; the majority of attack scenarios distribute across two or three verdict classes, reflecting the stochastic mechanisms of Section 6.4.3.

Three-regime comparison on discriminative features. Table 7.2.3 reports the mean and standard deviation of ten episode-level features under each of the four regimes. The features chosen are those that the downstream encoder either consumes directly (action counts, reject percentage, unique nodes) or derives at the step level (no-progress and consecutive-denied counters from Section 6.1.2); they are the hand-engineered counterpart of the learned representation evaluated in Section 7.3.

Table 7.2.3: Episode-level statistics by regime, reported as mean (std)

Feature	Normal ($n = 450$)	Successful ($n = 41$)	Partial ($n = 72$)	Blocked ($n = 37$)
env_steps	21.28 (11.56)	23.51 (10.40)	25.81 (15.95)	3.38 (3.96)
completion_ratio	0.77 (0.31)	0.96 (0.06)	0.70 (0.30)	0.08 (0.18)
reject_pct	0.38 (0.21)	0.32 (0.14)	0.68 (0.23)	0.49 (0.36)
unique_nodes	2.18 (0.76)	2.90 (0.89)	3.39 (1.21)	1.41 (0.76)
max_no_progress	6.77 (2.79)	5.49 (2.51)	7.94 (2.58)	10.27 (0.45)
max_consecutive_denied	1.18 (1.25)	1.00 (1.12)	1.93 (1.09)	1.89 (1.84)
n_navigate	19.24 (15.23)	36.12 (17.22)	27.62 (15.51)	5.73 (6.37)
n_invoke_tool	17.88 (14.13)	12.15 (9.54)	9.57 (11.67)	0.19 (0.52)
n_escalate	1.81 (7.57)	0.00 (0.00)	0.24 (0.72)	0.00 (0.00)
n_inspector	9.17 (8.06)	11.39 (10.50)	8.40 (7.22)	1.46 (1.17)

The table establishes three properties of the discriminative budget that the downstream model will work against.

No single feature separates all four regimes. The blocked regime is characterized by very short episodes (env_steps $= 3.38$) and near-zero completion (compl. $= 0.08$), and is well separated from the other three on those two features. The successful and partial regimes, however, overlap heavily with the normal regime on every individual feature: their means on `env_steps`, `reject_pct`, and `unique_nodes` all fall within one

standard deviation of the normal mean. This is the quantitative form of the geometric overlap visualized in Figure 7.2.10, and it is the empirical foundation for the argument that point-in-time, single-feature gating of the kind reviewed in Section 3.2.3 cannot achieve clean separation on this dataset.

The successful and partial regimes differ from normal in distribution shape, not in central tendency. On `n_navigate`, the successful regime has mean 36.1 versus 19.2 for normal—roughly twice as many navigation actions per episode—while the standard deviations are 17.2 and 15.2 respectively, so the distributions overlap substantially despite the shifted means. On `completion_ratio`, the successful regime concentrates very tightly around 0.96 (std 0.06) while normal scatters over the full $[0, 1]$ interval (std 0.31). These shape differences are exploitable by a density model—the CVAE assigns probability mass to the normal distribution and reports low likelihood for any region the normal distribution does not cover—but are invisible to a fixed-threshold classifier on any single feature.

The action mix is the most informative individual signal, but not in isolation. Across the four regimes, the $\langle$n_navigate, n_invoke_tool$\rangle$ pair takes distinct mean values (19.2/17.9 normal, 36.1/12.2 successful, 27.6/9.6 partial, 5.7/0.2 blocked), suggesting that a two-feature linear classifier could already separate the regimes at the level of class-conditional means. The class-conditional standard deviations, however, are larger than the inter-class mean gaps for the normal-vs-partial pair, so that classifier would have substantial overlap in practice. The behavioral-geometry projections in Section 7.2.3 confirm this: the linear PCA panel of Figure 7.2.10 shows partial overlap between exactly these two regimes.

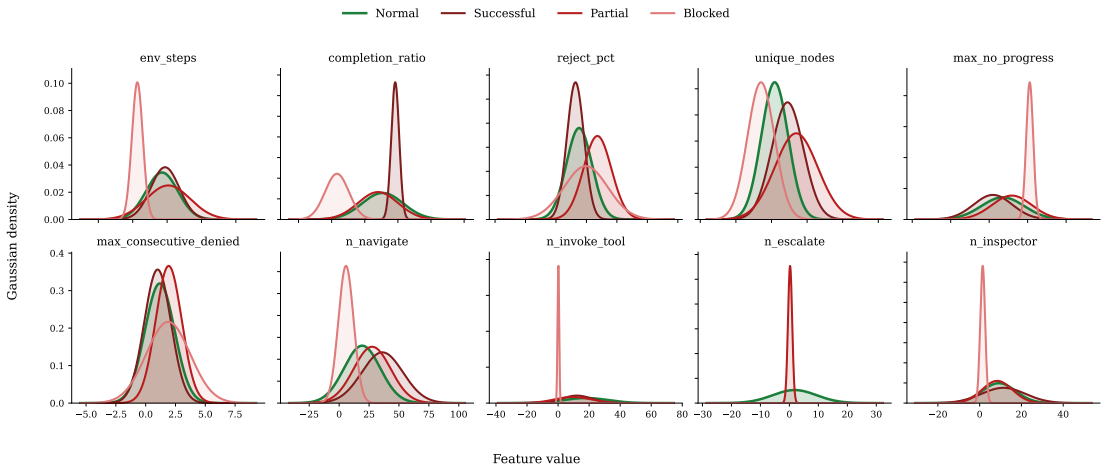

Figure 7.2.3: Visualization of Gaussian-distributed feature behavior across regimes. The subplots illustrate the separability and overlap structure that defines the discriminative budget available to the downstream model.

Taken together, the descriptive statistics in this subsection establish the dataset as one in which the four regimes are present, distinct in distribution, but not separable by any single hand-engineered feature. This is the regime in which a learned, temporal representation has something to recover, and it is the empirical setup for the representation evaluation in Section 7.3.

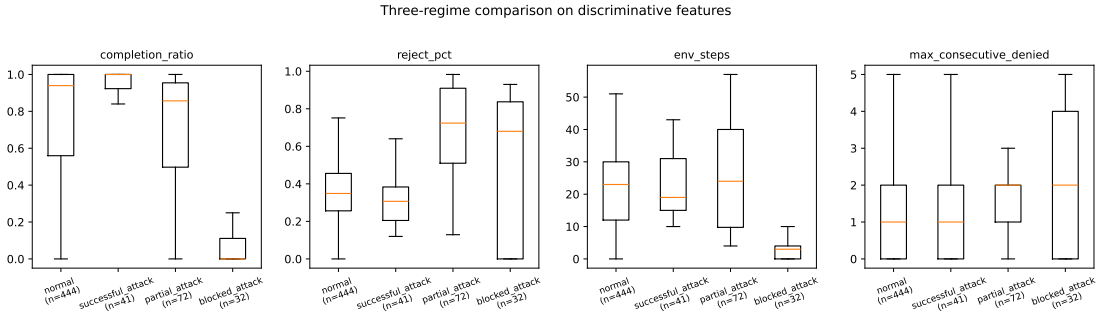

Figure 7.2.4: Distribution of four discriminative features across the four regimes (normal, successful attack, partial attack, blocked attack), shown as boxplots without outlier whiskers. Per-regime sample sizes are annotated on the x-axis. The blocked regime is sharply separated on completion ratio and episode length; the successful and partial regimes overlap with normal on every individual feature, motivating the multi-feature learned representation evaluated in Section 7.3.

7.2.2 Stream-Level Analysis

The episode-level descriptive statistics in Section 7.2.1 treat each episode as a single point. The CVAE+LSTM, however, consumes the trajectory at the per-step level: at each time step the encoder receives the raw observation vector and the action and auditor-decision identifiers from the previous step. The shape, range, redundancy, and discriminative content of those streams determine which channels can be included in the encoder input, which must be dropped for being constant, and which carry enough mutual information with the regime label to be worth carrying through the network. This subsection reports those four diagnostics in turn and ends with a sequence-level entropy analysis that establishes that the trajectory-level signal is non-trivial even after the per-step features have been characterized.

The unit of analysis throughout this subsection is the time step. After flattening every step from every episode (including those excluded by the integrity filter, since a degenerate stream would itself be a finding), the per-step table contains 92,482 rows. Each row carries 14 raw observation channels, 8 numeric derived features, 4 categorical identifiers, and the episode-level regime label.

Numeric streams. Table 7.2.4 reports the per-step mean, standard deviation, minimum, and maximum of the 22 numeric streams, computed over the 69,231 normal-data steps so that no information from the attack split influences the preprocessing decision.

Table 7.2.4: Per-step descriptive statistics of the numeric streams (normal split, 69,231 steps)

Feature	Mean	Std	Min	Max	Note
current_node_id	2.52	2.14	0.00	8.00	
resource_sensitivity	0.33	0.27	0.00	1.00	
last_action_status	0.11	0.37	0.00	2.00	
active_iam_role	0.00	0.00	0.00	0.00	constant; dropped
step_fraction	0.24	0.20	0.00	1.00	
visits_Public_Search	2.39	3.41	0.00	22.98	
visits_Help_Wiki	1.58	2.89	0.00	20.01	
visits_Calculator	0.92	2.59	0.00	23.02	
visits_Employee_Directory	1.41	2.82	0.00	23.02	
visits_Project_API	1.26	2.48	0.00	21.03	
visits_Knowledge_Base	0.62	2.03	0.00	22.97	
visits_Financial_Gateway	0.04	0.18	0.00	1.07	sparse
visits_Admin_Panel	0.03	0.15	0.00	1.07	sparse
visits_Production_DB	0.04	0.17	0.00	1.08	sparse
dt_from_prev	0.33	0.27	0.00	6.18	
no_progress_count	1.98	1.87	0.00	11.00	
consecutive_denied	0.11	0.44	0.00	5.00	
consecutive_inspections	0.48	0.72	0.00	2.00	
rejection_count	1.12	3.01	0.00	51.00	heavy-tailed
cpu_percent	39.43	215.50	0.00	4201.60	noisy outliers
ram_mb	0.00	0.00	0.00	0.00	constant; dropped
latency_ms	0.21	0.08	0.09	0.63	

Three observations from this table fix the encoder input vector.

First, two streams are exactly constant on the normal split: `active_iam_role` (always 0) and `ram_mb` (always 0). Both reflect environment instrumentation that was specified in the schema but never wired in the data-collection pipeline of Section 6.4.1: the IAM role channel was reserved for the future controller output, not for the environment-side observation, and the RAM telemetry hook was never connected. A constant feature contributes no variance for the encoder to model and no information for the decoder to reconstruct, so both channels are dropped before tensor export, leaving 12 raw observation dimensions plus 6 active numeric derived features.

Second, two streams require attention before standardization. The `cpu_percent` stream has a standard deviation of 215.5 against a mean of 39.4 and a maximum of 4,201.6, which is more than 19 standard deviations above the mean — the signature of a small number of extreme outliers from the host operating system rather than meaningful behavioral signal. The `rejection_count` stream is heavy-tailed in a different and informative way: its maximum of 51 corresponds to the N08 escalation scenario, where the auditor legitimately refuses many proposed escalations, and this tail is therefore real signal that must be preserved. The two streams are accordingly

treated differently in preprocessing: `cpu_percent` is z-scored (normalized) along with the other continuous features but is flagged as a low-utility candidate in the mutual-information ranking below, while `rejection_count` is preserved with its full dynamic range (no normalization due to the fact that it is a strong signal).

Third, the visit-counter block exhibits a clear sensitivity gradient that reflects the zone structure of Figure 6.1.1. The Zone 1 and Zone 2 nodes (`Public_Search` through `Knowledge_Base`) carry means between 0.6 and 2.4 with maxima above 20, indicating routine traffic.

The three Zone 3 nodes (`Financial_Gateway`, `Admin_Panel`, `Production_DB`) are an order of magnitude sparser, with means below 0.05 and maxima just above 1.0. This sparsity is by design — Zone 3 access is rare in the benign distribution — and is the property that the CVAE will learn to reproduce as the normative pattern. Frequent visits to Zone 3 nodes in an attack episode therefore translate to a localized region of the input space the decoder has not been trained to reconstruct, which is the operating principle behind the reconstruction-error signal in Section 7.3.

Redundancy among the raw observation channels. In the 14-dimensional input the visit-counter channels are not guaranteed to be independent: an episode that spends time in Zone 2 may visit several Zone 2 nodes in sequence, producing partial correlation between their visit counts. The Pearson correlation matrix of the 12 non-constant raw-observation streams (computed on the normal split) shows a block structure consistent with the zone partition: within-zone visit pairs are weakly positively correlated ($\rho \in [0.1, 0.3]$ for Zone 1 and Zone 2 pairs), while between-zone visit pairs are uncorrelated or weakly negatively correlated, the latter reflecting the fact that an episode that dwells in one zone has fewer remaining steps to spend in another. `step_fraction` correlates positively with all visit counts, as it must mechanically (all four quantities advance with elapsed time). No correlation exceeds 0.6 in absolute value, so no channel is redundant in the strong sense that would justify dropping it.

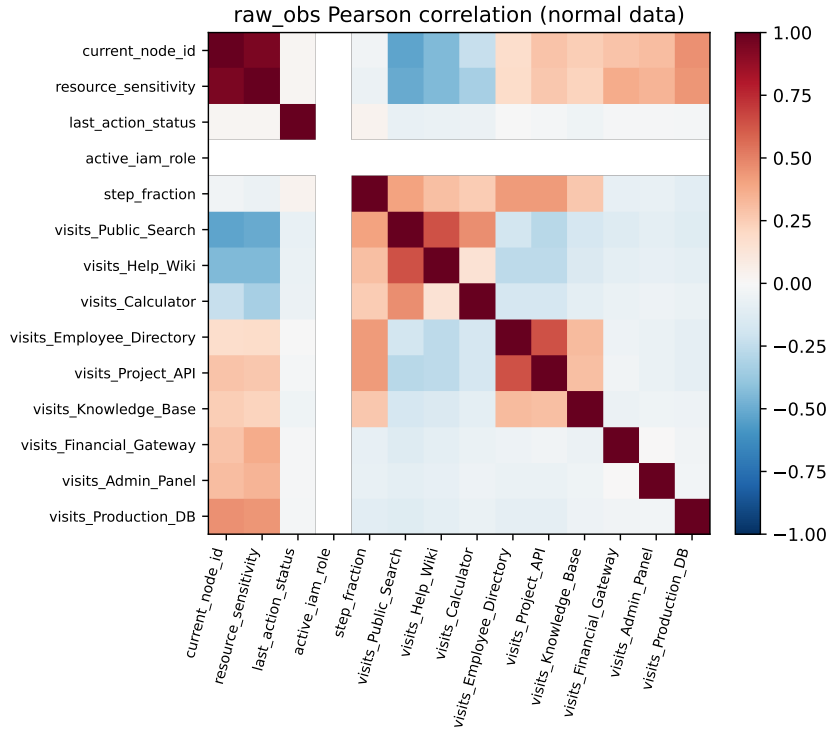

Figure 7.2.5: Pearson correlation matrix of the 14 raw observation channels computed on the normal split. Within-zone visit-counter pairs are weakly positively correlated; between-zone pairs are uncorrelated or weakly negatively correlated. No correlation exceeds 0.6 in absolute value, ruling out strong-sense redundancy among the observation channels.

The PCA scree on the standardized 14-channel matrix confirms this diagnosis: 10 principal components are required to capture 95% of the variance, and 12 are required for 99%. With only two of the 14 nominal dimensions being constant — and therefore contributing zero variance — the effective dimensionality of the raw observation block is essentially its full 12-channel rank. This rules out a low-rank linear bottleneck as a viable encoder design and supports the choice of a latent dimension at least as large as the number of independent raw-observation channels.

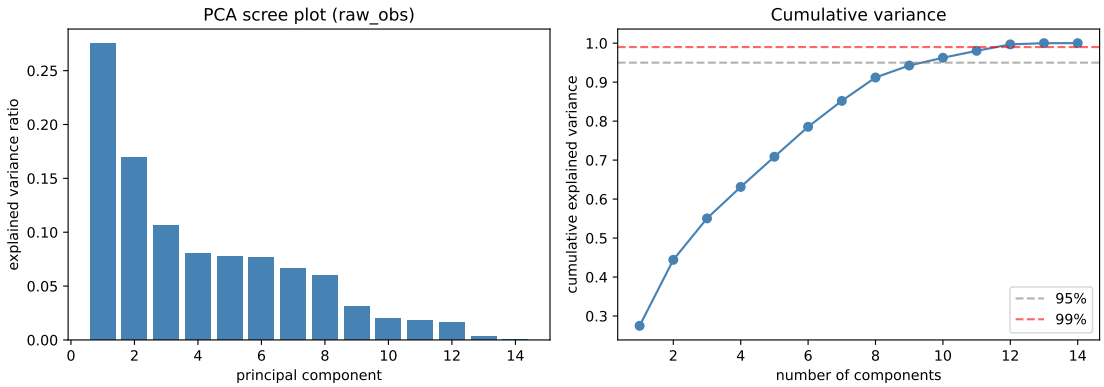

Figure 7.2.6: PCA on the standardized 14-channel raw observation block. Left: explained-variance ratio per principal component. Right: cumulative explained variance with the 95 % and 99 % thresholds marked. Ten components are required for 95 %, twelve for 99 %; with two of the 14 channels being constant (Section 7.2.2), the effective rank is essentially full.

Categorical streams. Four categorical streams accompany the numeric block: `action_type` (the action class issued by the controller), `action_target` (the resource node or tool the action operated on), `auditor_decision` (approve or reject), and `short_id` (the scenario identifier of the episode). Their distributions on the normal split are summarized in Table 7.2.5 via Shannon entropy and Simpson diversity. Normalized entropy near 1.0 indicates a near-uniform distribution; normalized entropy near 0 indicates one category dominates and the feature is effectively constant.

Table 7.2.5: Categorical stream diversity on the normal split

Feature	$\lvert\mathcal{V}\rvert$	Top value	Top frac.	H (bits)	$H/H_{\max}$	Simpson
action_type	6	navigate	0.398	1.759	0.681	0.669
action_target	15	read	0.307	3.027	0.775	0.843
auditor_decision	2	approve	0.624	0.955	0.955	0.469
short_id	15	N10	0.085	3.831	0.981	0.928

The four streams sit at three different operating points. `short_id` is near-uniform ($H/H_{\max} = 0.98$) by construction: each of the 15 normal scenarios contributes 30 episodes, and the per-step counts inherit roughly that balance up to the per-scenario length variation documented in Table 7.2.1. This serves as a sanity check that no scenario silently dominates the training distribution. `action_target` has 15 distinct values ($H/H_{\max} = 0.78$); `read` is the most common at 30.7% but no single target dominates, which confirms that the controller exercises the full tool inventory rather than collapsing to one operation. `auditor_decision` is the most uniform of the four ($H/H_{\max} = 0.96$): the auditor approves 62.4% of proposals on benign data, which translates to a 0.624/0.376 split that is close to the maximum-entropy point for a binary variable. This is the per-step counterpart of the per-episode `reject_pct` variation reported in Section 7.2.1, and it confirms that auditor rejection on the benign distribution is frequent enough to be a noisy rather than a clean signal.

`action_type` is the only categorical stream whose distribution is meaningfully concentrated, with `navigate` accounting for 39.8% of all controller decisions. This is consistent with the environment design — navigation between resource nodes is the prerequisite for almost every other action — and the 0.681 normalized entropy still leaves enough mass on the remaining five action classes that the feature is informative rather than degenerate.

Sequence-level structure. The per-step categorical analysis treats each step in isolation. The controller, however, is a sequential decision-maker, and the order in which actions occur is precisely the structure the LSTM component of Section 5.2 is intended to capture. Two sequence-level diagnostics establish that this structure is present in the data and is not dominated by trivial repetition.
First, of the 600 episode-level action sequences, 563 are unique (93.8%). Despite 30 repetitions of each normal scenario and 10 repetitions of each attack scenario under the parameterized template distribution, only 37 episodes share an exact action sequence with any other episode. This rules out the failure mode in which the

diversity mechanisms of Section 6.4.3 produce nominally distinct episodes that all collapse to the same trajectory.

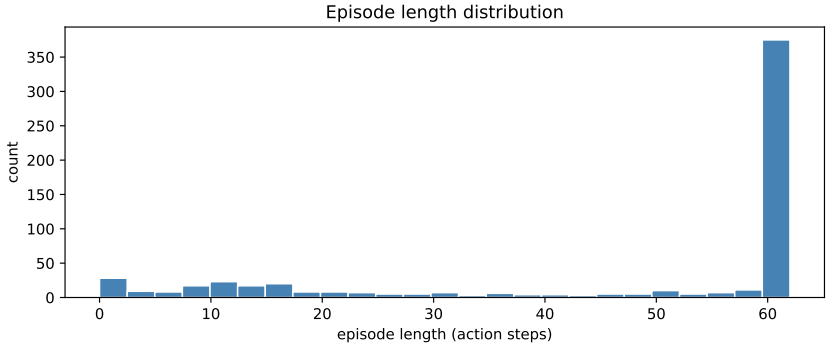

Figure 7.2.7: Histogram of episode length in action steps across all 600 episodes. The distribution is right-skewed with a boundary concentration at the maximum episode length, reflecting both long-horizon scenarios (N09, N10) and episodes terminating at the step limit.

Discriminative utility of the streams. Second, the n-gram entropy of the action sequences increases monotonically with n: the unigram entropy is 1.69 bits, the bigram entropy is 3.13 bits, and the trigram entropy is 4.46 bits. The unigram value is the entropy of the marginal action distribution and matches the `action_type` entropy in Table 7.2.5 up to the difference between controller actions and the broader action-target alphabet. The strict increase from unigram to bigram to trigram confirms that the sequence of actions carries information beyond what the marginal distribution provides — i.e., the data has temporal structure that a memoryless encoder cannot recover. If the trigram entropy had matched the unigram value at $1.69 \times 3 = 5.07$ bits, the sequences would have been indistinguishable from independent action samples and the LSTM backbone would have been unmotivated; if instead the trigram entropy had collapsed toward the unigram value, the sequences would have been near-deterministic and the recurrent capacity would have been unnecessary. The observed intermediate value is the regime in which a recurrent encoder has something to learn.

The stream-level statistics so far describe the data's marginal properties without reference to the regime label. A complementary diagnostic asks how much information each stream individually carries about the binary normal-vs-attack distinction, estimated by per-feature mutual information against the binary label. The features that exceed the 0.05-bit retention threshold are dominated by the guard-counter block — `rejection_count`, `no_progress_count`, `consecutive_denied` — together with the visit counters for the Zone 3 nodes (`visits_Financial_Gateway`, `visits_Admin_Panel`, `visits_Production_DB`), which are sparse on benign data and therefore provide a strong asymmetric signal when an attack episode visits them. `cpu_percent` and `latency_ms` fall below the threshold, consistent with the heavy-tailed-noise diagnosis above for the former and with the very narrow operating range $[0.09, 0.63]$ for the latter; both are nevertheless retained in the encoder input on the grounds that mutual information against a binary label penalizes features that are informative only conditional on context, which is precisely what a recurrent encoder is designed to exploit.

The mutual-information ranking is therefore not used as a hard feature selector but as a sanity check on the input vector composition: no retained feature is below the

threshold for an unrelated reason, and no high-MI feature has been inadvertently dropped.

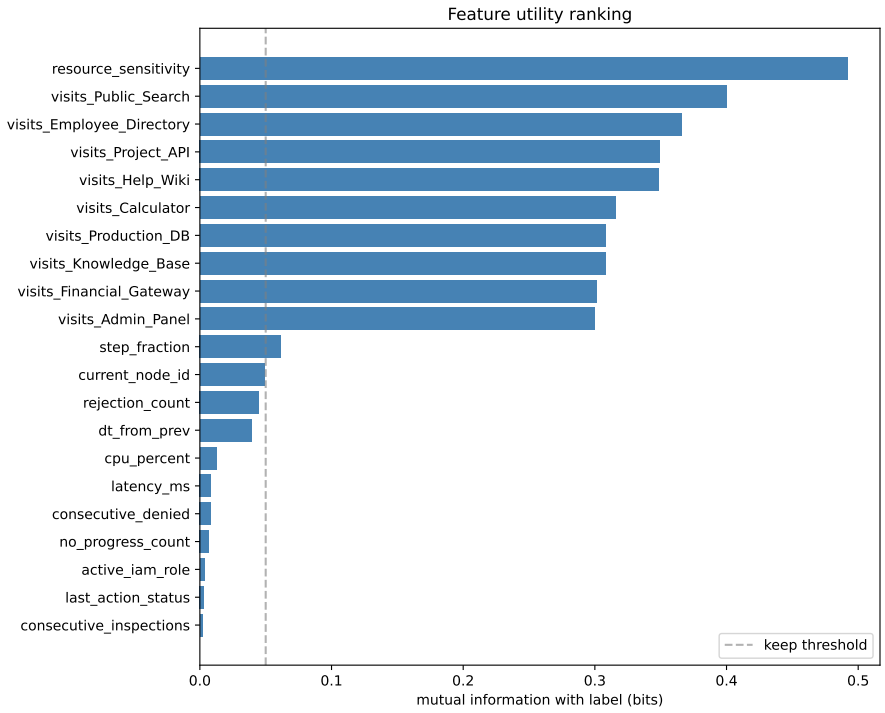

Figure 7.2.8: Per-feature mutual information against the binary normal-vs-attack label, computed on the 92,482 per-step rows. Features are ordered by mutual information; the dashed line marks the 0.05-bit retention threshold. The guard-counter block (`rejection_count`, `no_progress_count`, `consecutive_denied`) and the Zone 3 visit counters dominate the ranking; `cpu_percent` and `latency_ms` fall below the threshold but are retained in the encoder input for context-conditional utility.

Per-batch drift. The dataset was generated in three normal batches and two attack batches, each batch corresponding to a fixed combination of seed, `tool_fail_prob`, `obs_noise_std`, and `edge_disable_count`; the parameter triple is recorded in `episode_config` and uniquely identifies the generating run. To check whether the parameter perturbations introduced systematic distributional drift, we ran two-sample Kolmogorov–Smirnov tests on six episode-level summaries (`env_steps`, `completion_ratio`, `reject_pct`, `tool_invocations`, `unique_nodes`, and the `invoke_tool` action share) and on the per-step controller-action interval `dt_from_prev`, applying a Bonferroni correction across pairs within each feature.

The combined picture is that the parameterisation produced the small, controlled diversity it was designed for—visible only as a mild gradient on `completion_ratio` that tracks the `tool_fail_prob` setting across the three normal batches—without inducing systematic drift on the channels the encoder consumes. Figure 7.2.9 illustrates this for the per-step controller interval `dt_from_prev`: the empirical CDFs of all three normal batches and both attack batches lie on top of each other, with a worst-case $D \leq 0.04$ in either regime.

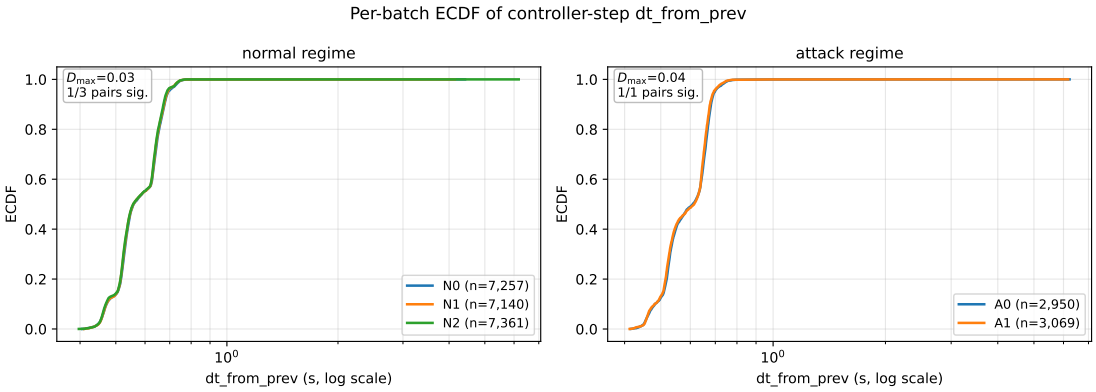

Figure 7.2.9: Per-batch empirical CDF of the per-step controller interval `dt_from_prev` (log-scale x-axis), normal regime on the left, attack regime on the right. Curves coincide across batches: $D_{\max} \leq 0.04$ in both regimes. The "significant" pairs flagged in each panel are an artefact of the large per-step sample size ($n \approx 7000$ per batch) and do not correspond to any operationally meaningful drift.

Metadata leakage. A separate concern is whether the episode-level metadata (features that a deployed system would have access to but that are not included in the encoder input) already carries the normal-vs-attack label. If it does, the CVAE evaluation would be inflated by a shortcut the model never actually sees at inference. To quantify this, a logistic regression was trained on seven episode-level metadata summaries (`n_steps`, `env_steps`, `successful_actions`, `denied_actions`, `tool_invocations`, `unique_nodes`, `zone_crossings`) under 5-fold cross-validation with balanced class weights, against the binary normal-vs-attack label. The resulting ROC-AUC is 0.645 ± 0.127, well below the 0.9 threshold that would flag trivial recoverability and only modestly above the 0.5 chance baseline. The 5-fold spread of ± 0.127 reflects the asymmetric class sizes (450 normal versus 150 attack) rather than instability of the classifier. The interpretation is that some coarse-grained metadata signal does exist (e.g. attacks run for different durations than benign scenarios on average) but not enough of it to short-circuit the evaluation: the encoder still has to recover the regime structure from the per-step behavioural streams, not from episode-level summary statistics.

Encoder input vector. The four diagnostics above jointly determine the per-step input vector the CVAE+LSTM consumes. After dropping the two constant raw-observation channels, the active block contains 12 raw observation features. Six of the eight numeric derived features are retained — the guard counters `no_progress_count`, `consecutive_denied`, `consecutive_inspections`, `rejection_count`, plus `dt_from_prev` and the two host-telemetry channels `cpu_percent` and `latency_ms` retained for context-conditional utility — yielding a 19-dimensional continuous block that is z-scored using statistics fit on the normal split only. Two categorical identifiers (`action_type_id` and `auditor_decision_id`) are appended without scaling, producing a final per-step input dimensionality of 21. All preprocessing statistics are fit on the normal split exclusively, so no information from the attack episodes leaks into the standardization, which is the prerequisite for the leakage check above.

7.2.3 Behavioral Geometry — Low-Dimensional Projections

Before training the CVAE+LSTM described in Section 5.2, it is useful to verify that the behavioral dataset collected under the procedure of Section 6.4 exposes structure that a learned temporal model can exploit. If the four-regime decomposition defined in Section 4.4.3 —*normal*, *successful attack*, *partial attack*, *blocked attack*—is visible in simple, hand-engineered summary features, then a trivial classifier would already suffice and the temporal architecture proposed in Chapter 5 would be unmotivated. Conversely, if no regime structure is visible at any scale, then no model, regardless of capacity, can recover the signal from this data. The projections presented in this section occupy the middle ground: they test, visually, at what scale the regime structure becomes discernible, and they thereby provide empirical grounding for the architectural choices made in Chapter 5.

Six projections are produced, spanning three temporal scales (episode-level, window-level, and step-level) and two labeling axes (four-regime and per-scenario). All projections use PCA, t-SNE, and UMAP on z-scored feature matrices; the same projection is colored and marker-encoded under different labelings so that visual differences reflect labeling rather than geometry. To support readers with color-vision deficiencies and to ensure readability in monochrome print, each regime is redundantly encoded by both color and marker shape (normal: blue circles; successful attack: red triangles; blocked attack: orange squares; partial attack: purple diamonds). This redundant encoding is applied consistently across all projections in this section.

Episode-Level Regime Separation

Figure 7.2.10 projects each valid episode as a single point in an 11-dimensional summary vector consisting of navigation counts, action-type fractions, auditor reject pressure, guard-counter maxima, and progress statistics (these are the most relevant features that static risk-based access control models would rely on for scoring). Three projections of this matrix are shown side by side: PCA, t-SNE, and UMAP.

This figure is the best test of the argument made in Section 3.4 regarding the limitations of static feature-based risk estimation. If the four regimes cleanly partition into disjoint clouds in the linear PCA projection, the entire case for a learned temporal representation collapses (a simple logistic regression on the 11 summary features would be sufficient, and the complex architecture of Chapter 5 would be unjustified). The observed overlap between regimes in the PCA panel, contrasted with the partial separation visible in t-SNE and UMAP, is therefore an evidence in support of the thesis claim that the regime boundary is not a linear hyperplane in the summary-statistics space, and hand-engineered rules do not capture the full discriminative signal.

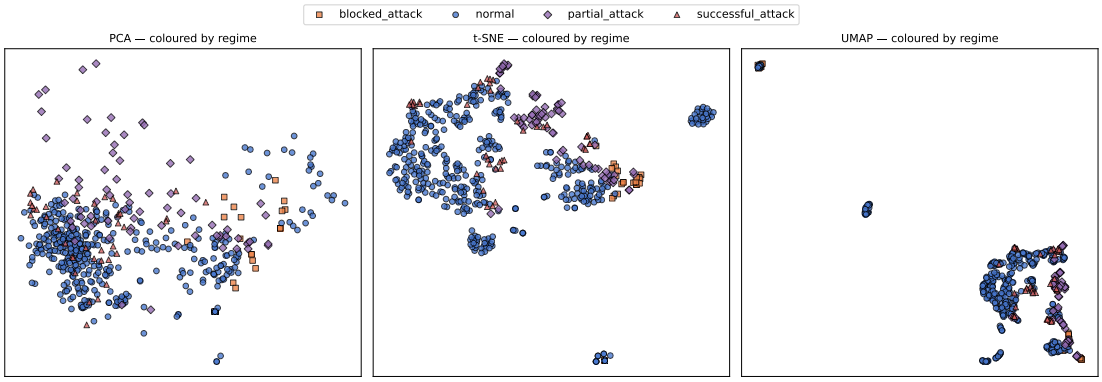

Figure 7.2.10: Episode-level regime separation under PCA, t-SNE, and UMAP. Each point is one valid episode, represented by an 11-dimensional fingerprint of summary statistics. Regimes are encoded redundantly by color and marker shape: normal (blue circles), successful attack (red triangles), partial attack (purple diamonds), blocked attack (orange squares).

Per-Scenario Behavioral Diversity — Normal Scenarios

Figure 7.2.11 evaluates whether the scenario parameterization and stochastic diversity mechanisms described in Section 6.4.3 produce genuinely distinct behavioral distributions. The projection is the UMAP embedding from Figure 7.2.10. In each panel, the highlighted scenario is drawn in blue over a grey background consisting of all other episodes (both normal and adversarial).
Two properties are informative here. First, *within-scenario compactness*: highlighted points should form a recognisable cluster rather than scatter randomly, indicating that the scenario's semantic identity translates into a consistent behavioral signature. Second, *between-scenario displacement*: different scenarios should occupy different regions of the embedding, indicating that the scenario library from Section 6.2.2 covers a meaningful range of the behavioral space rather than collapsing onto a dominant mode. The joint satisfaction of both properties supports the claim that the 15 normal scenarios sample a behaviorally diverse baseline against which deviation can be meaningfully measured.

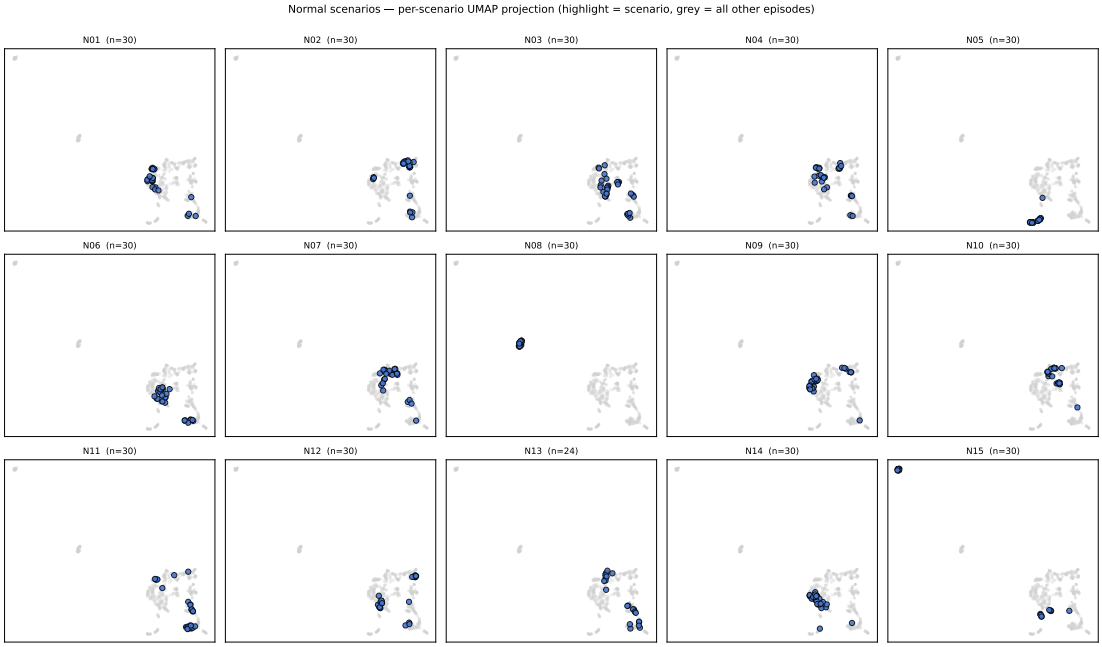

Figure 7.2.11: Per-scenario UMAP projection of normal scenarios, shown as fifteen small multiples. In each panel the highlighted scenario is rendered in blue and all remaining episodes (normal and adversarial) are rendered in grey. The n in each title is the number of episodes available for that scenario after integrity filtering.

Per-Scenario Attack Localization

Figure 7.2.12 repeats the small-multiples construction for the attack scenarios, highlighting each in red. It serves a different analytical purpose than the normal counterpart. As argued in Section 4.4.3, the attack set is deliberately broader than the three structural classes (goal hijacking, memory poisoning, cascading failures) that the framework is designed to address. The per-scenario panels allow the reader to assess, attack by attack, whether the attack's behavioral signature localizes to a recognisable pocket of the embedding.

Three qualitative outcomes are expected and each carries a distinct interpretation. Attacks whose behavioral signature is mechanically strong (e.g denial-of-service loops) should localize to tight off-cloud pockets, as their trajectories are quantitatively unlike any benign scenario. Attacks that diverge from normal behavior only weakly (e.g. indirect prompt injection that produces a single mis-targeted invocation that looks behaviorally like a legitimate sequence of reads) are expected to overlap the normal cloud; this overlap is precisely the regime in which static summary statistics fail and a learned representation becomes necessary. Finally, some attacks may split across multiple regions, this corresponds to the verdict heterogeneity established in Section 7.2.3: a single attack may yield successful, partial, and blocked outcomes depending on stochastic factors, and this manifests geometrically as multi-modal scatter.

Overall, the framework is trained and evaluated against a broader attack set (than the three classes it is theoretically designed to detect), and the per-scenario projections make the boundary between in-scope and out-of-scope attacks geometrically visible.

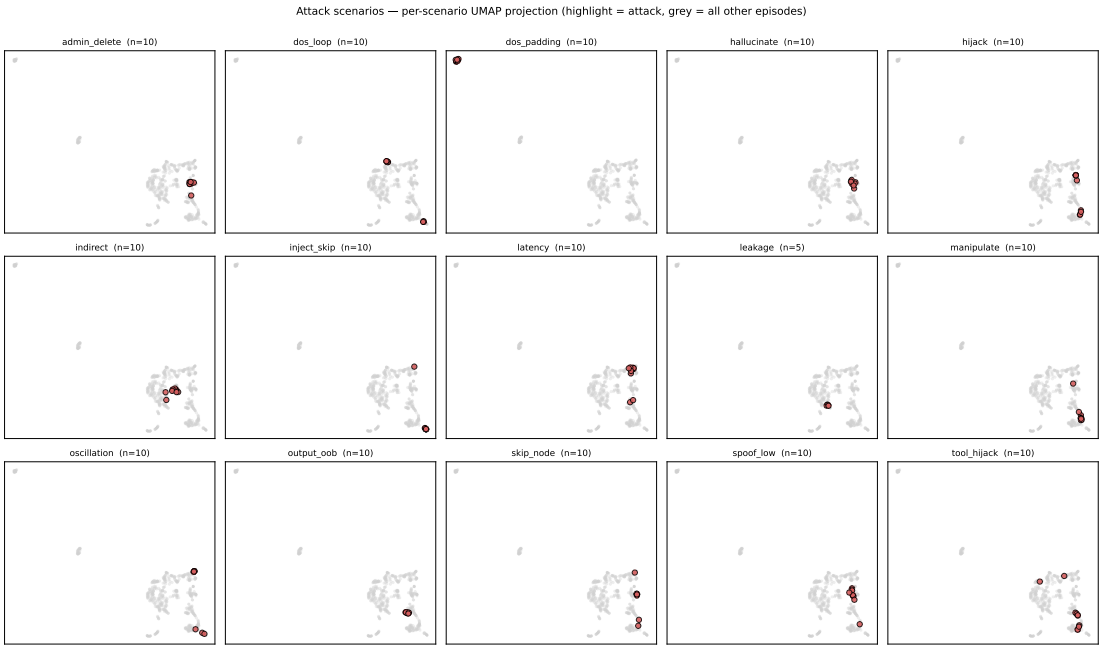

Figure 7.2.12: Per-scenario UMAP projection of attack scenarios, shown as fifteen small multiples. In each panel the highlighted attack is rendered in red and all remaining episodes (normal and adversarial) are rendered in grey. Tight off-cloud highlights correspond to attacks whose behavioral signature is mechanically strong; scatter overlapping the grey cloud corresponds to attacks whose signature is subtle at the episode-summary level.

Per-Step Ambiguity — the Pre-Temporal Baseline

Figure 7.2.13 projects a balanced sample of individual time steps drawn from the three regimes, colored by regime label. The feature vector is the raw observation plus the guard counters defined in Section 6.1.2, and is exactly the input a non-recurrent encoder would receive at each decision point. This figure serves as the lower bound in the temporal-scale sweep: it shows what is visible without any temporal context. The expected and informative outcome is that per-step projections show heavier regime overlap than per-episode projections. A single step in a successful attack is frequently locally indistinguishable from a step in a legitimate scenario, because both use the same observation schema and invoke the same underlying tools; what differs is the sequence in which the steps occur and the cumulative state they produce. If the per-step plot already exhibited clean regime separation, the CVAE+LSTM's recurrent backbone would be redundant and a simple memoryless encoder would suffice. The observed overlap therefore constitutes direct empirical support for the LSTM component (the signal is not present at the single-step level, and must be recovered from temporal accumulation).

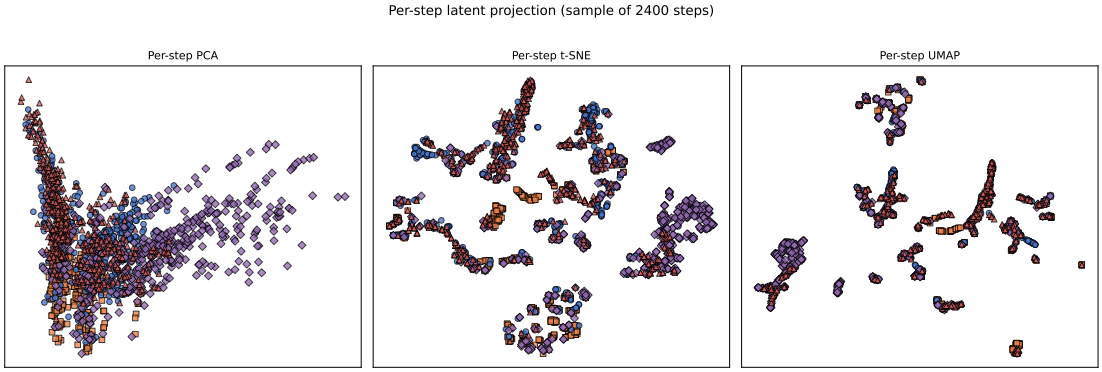

Figure 7.2.13: Per-step projection under PCA, t-SNE, and UMAP, on a balanced sample of individual time steps drawn from the three regimes and colored by regime label. Per-step regime overlap is heavier than per-episode overlap (Figure 7.2.10), confirming that single-step features are insufficient for regime discrimination and motivating the recurrent backbone of Section 5.2.

Rolling-Window Projection — the LSTM's Receptive Field

Figure 7.2.14 occupies the intermediate scale between Figure 7.2.13 (per-step) and Figure 7.2.10 (per-episode). For each episode, a sliding window of length $W = 5$ steps is extracted with stride 2; each window is summarised by the concatenation of per-feature means and standard deviations, producing a feature vector of twice the step-level dimensionality.

The relative separation quality across Figure 7.2.13 (per-step), Figure 7.2.14 ($W = 5$ windows), and Figure 7.2.10 (per-episode) is the most direct empirical statement available at this stage about the CVAE+LSTM architectural choice. Three orderings are possible and each has a distinct interpretation: (i) if per-step and window-level separation are comparable, the temporal dimension contributes little and a non-recurrent encoder would suffice; (ii) if separation grows monotonically from per-step through window-level to per-episode, the signal accumulates with temporal context in a manner the LSTM's hidden state can be expected to capture; (iii) if the window-level already matches the per-episode separation, the signal saturates within a short horizon and the LSTM's effective receptive field need not extend to the full episode length. Here by comparing the three plots, it is obvious that the separation is growing but not saturated, and this is the strongest motivation for the LSTM architecture.

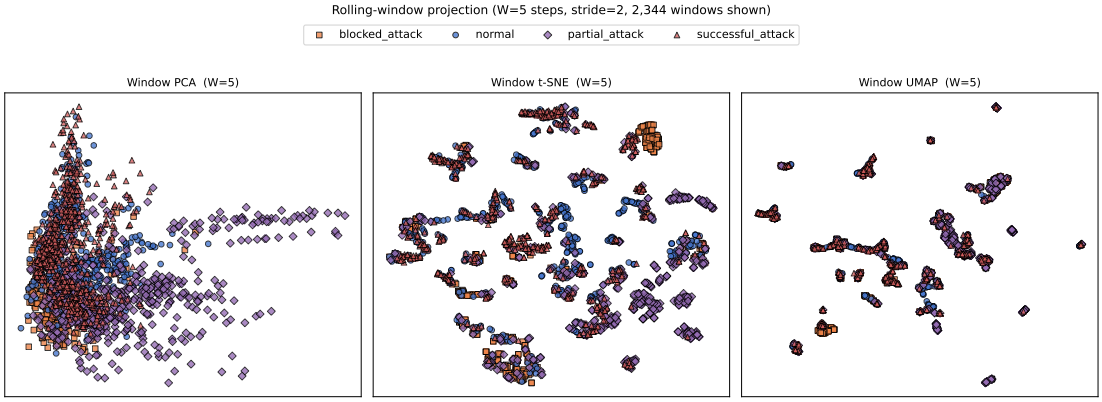

Figure 7.2.14: Rolling-window projection under PCA, t-SNE, and UMAP, on fixed-length windows of $W = 5$ consecutive time steps with stride 2. Each point is the concatenated mean and standard deviation of the per-step features over one window, which approximates the intermediate-horizon information available to the LSTM hidden state. Regime separation at this scale lies between the per-step and per-episode views, providing direct empirical support for the recurrent encoder choice.

Window-Size Sweep — Horizon of the Signal

Figure 7.2.15 extends the previous construction by sweeping the window length across $W \in \{1, 3, 5, 10, 15\}$, with all other parameters held constant.
The figure operationalizes the research-methodological question that asks how much temporal context is needed to separate regimes. Here although the pattern is not clear enough but it seems like there exist a monotonic improvement in regime separation but by increasing th W it is reaching to saturation. This can be a further motivation for the choice of LSTM, but at this point we cannot make a stronger argument.
The practical relevance of this sweep extends beyond visual analysis. The selected value of W^* (at which we reach convergence) can inform the truncated backpropagation-through-time length during CVAE+LSTM training (Section 7.3.1), and can serve as a prior for the LSTM's hidden-state dimensionality: a short horizon argues for a compact hidden state, while a long horizon justifies the capacity overhead of a larger one.

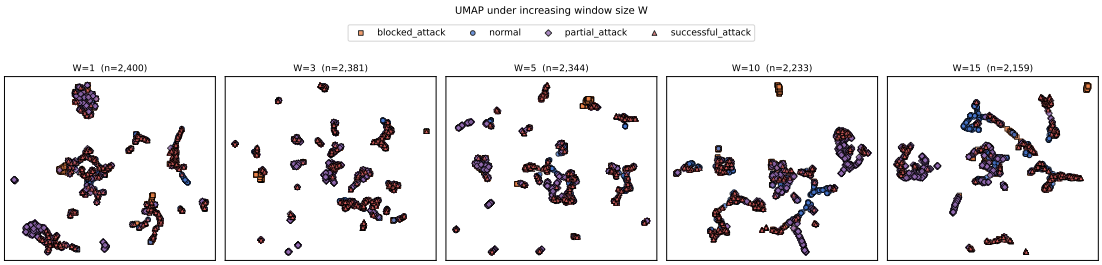

Figure 7.2.15: Window-size sweep under UMAP, varying $W \in \{1, 3, 5, 10, 15\}$. Regime separation increases monotonically with W up to a saturation point, providing an empirical lower bound on the temporal horizon over which the regime signal accumulates and informing the receptive-field choice for the LSTM in Section 5.2.

Summary

The six projections presented in this section collectively address three distinct thesis claims. Figure 7.2.10 (episode regime separation) grounds the argument in

Section 3.4 against static-feature access control. Figures 7.2.11 and 7.2.12 validate the diversity mechanisms of Section 6.4.3 and visualize the in-scope/out-of-scope boundary articulated in Section 4.4.3. Figures 7.2.13, 7.2.14, and 7.2.15 (per-step, window, and sweep) operationalize the CVAE+LSTM choice of Section 5.2 by showing the temporal scale at which the regime signal emerges. In combination, they establish that the collected dataset is suitable for training a temporally-conditioned behavioral model and that such a model is necessary (neither a static classifier on summary features nor a memoryless encoder on individual steps would be sufficient). The quantitative verification of this claim, via reconstruction error and downstream detection performance, is the subject of Section 7.3.

7.2.4 Summary of Data Readiness

The three diagnostics of Sections 7.2.1–7.2.3 jointly establish that the collected corpus is suitable for training and evaluating the CVAE+LSTM of Chapter 5.
Composition. 589 of 600 episodes (98.2%) pass the integrity filter and decompose into 444 normal, 41 successful, 72 partial, and 32 blocked episodes; every regime exceeds the sample threshold.
Preprocessing. Two constant channels are dropped, leaving a 21-dimensional per-step input; standardisation is fit on the normal split only, and the metadata-leakage probe of Section 7.2.2 returns ROC-AUC 0.645, ruling out a trivial label shortcut.
Temporal structure. Per-step regime overlap is heavy and narrows monotonically with window size, which rules out a memoryless encoder and motivates the recurrent backbone of Section 5.2.
The residual limitations — 10 episodes per attack type, the simulator-to-field gap, and LLM-sampling non-determinism — are revisited in Section 8.2. The quantitative verification that the model recovers the signal identified here is the subject of Section 7.3.

7.3 Behavioral Representation — RQ1

This section reports the quantitative evaluation of the CVAE+LSTM behavioural encoder specified in Chapter 5. The dataset-readiness argument of Section 7.2 established that the four-regime signal is present in the data, that it emerges under temporal aggregation rather than at the single-step level, and that no metadata shortcut reduces the detection problem to a trivial lookup. The present section tests whether the trained model actually recovers that signal, how cleanly the reconstruction-error score calibrates to a usable risk signal, and which attack classes the model detects or misses. The section closes with an ablation that quantifies how much of the headline performance is carried by the nine per-node visit counters identified as a potential coverage-bias source in Section 7.2.2, and with a discussion of the failure modes that jointly motivate the historical-trust layer introduced in Section 7.4.

7.3.1 Training Setup

Two variants of the CVAE+LSTM were trained on the 310 valid normal episodes (training episodes): a baseline with a standard-Normal prior $p(z) = \mathcal{N}(0, I)$, and a

variant with an eight-component Gaussian mixture prior $p(z) = \sum_{k=1}^{8} \pi_k \mathcal{N}(\mu_k, \Sigma_k)$ with learnable mixture parameters. The two priors are compared head-to-head in Section 7.3.6. All other hyperparameters are held constant: latent dimension $d_z = 16$, LSTM hidden size 64, window length $W = 32$ with training stride 8, batch size 128, Adam optimiser with learning rate 10^{-3}, and a KL-warmup schedule that linearly anneals β from 0.1 to 1.0 over the first ten epochs. The continuous-feature reconstruction term uses a masked sum of squared errors over the 19 continuous features; the two categorical heads (`action_type`, `auditor_decision`) use cross-entropy.

The training/validation/test split partitions the 444 normal episodes at episode granularity (approximately 70/15/15%, seeded), yielding $6{,}129$ training windows, $1{,}371$ validation windows after tiling, and $1{,}396$ test windows. Standardisation statistics for the continuous block were fit exclusively on the training windows, consistent with the leakage argument of Section 7.2.2. The attack split plays no role in training and is never seen by the optimiser.

Training was run for 300 epochs with best-model selection on validation ELBO. The budget was chosen after two shorter runs (60 and 150 epochs) showed the best validation ELBO still being hit within the last two epochs, indicating insufficient convergence. At 300 epochs the best validation ELBO lands at epoch 287 for the standard-Normal prior and epoch 284 for the GMM prior, leaving a margin of more than ten epochs of further training that did not improve the score. The best validation objective (on the validation set) is 1.0039 for the standard-Normal prior and 1.0217 for the GMM prior, with train–validation gaps of 0.29 and 0.29 respectively — small relative to the absolute loss and consistent with no meaningful overfitting. Training curves for both models are shown in Figure 7.3.1.

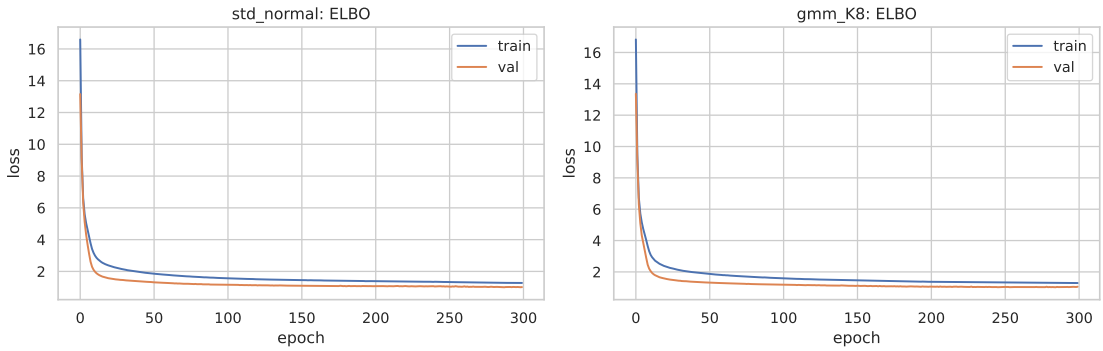

Figure 7.3.1: Training and validation objective for the standard-Normal and GMM-prior variants over 300 epochs. Both variants exhibit monotonically decreasing validation loss through training, with no late-epoch divergence between training and validation curves. The GMM-prior model attains a slightly higher validation objective, indicating that the additional prior flexibility does not improve reconstruction of the normal distribution.

The representation models were trained without attack episodes. However, the choice of the GMM-prior model for downstream analysis followed an exploratory comparison on primary-regime ROC–AUC. In addition, the selected logistic calibrator used labelled normal and attack episodes. The subsequent trust and controller results should therefore be interpreted as retrospective full-corpus analyses rather than untouched held-out generalisation tests.

7.3.2 Reconstruction Quality on Normal Data

The trained encoder's reconstruction error on normal validation data is the reference distribution against which all attack-side detection claims are measured. The per-episode mean reconstruction error, computed by tiling each episode into $W = 32$ windows with stride 16 and averaging the per-window error across valid timesteps, has median $\rho = 14.2$ and 95th percentile $\rho = 48.3$ on the normal validation split. Figure 7.3.2 contrasts the normal distribution with each of the three attack regimes. The normal boxplot is tightly packed near zero; the partial-attack boxplot has median $\rho \approx 450$, a thirtyfold separation from the normal median, with an interquartile range spanning three orders of magnitude; the blocked-attack boxplot sits between the two; and the successful-attack boxplot, strikingly, overlaps substantially with the normal distribution.

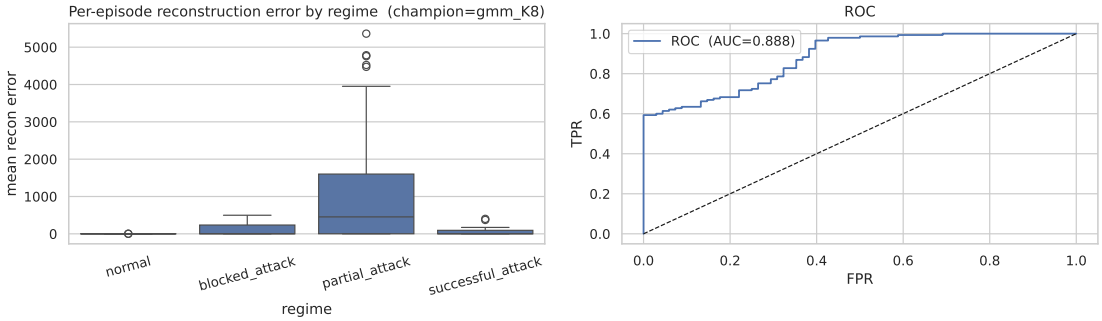

Figure 7.3.2: Per-episode mean reconstruction error ρ by regime (champion: GMM-prior CVAE+LSTM). Partial attacks separate cleanly from normal, blocked attacks separate moderately, and successful attacks overlap the normal distribution at the episode-mean granularity. This overlap is the empirical source of the differential detection difficulty reported in Section 7.3.4 and is the primary motivation for the historical-trust layer of Section 7.4.

The successful-attack overlap is the single most consequential observation in this section and merits explanation before any detection number is reported. A successful attack, by the definition in Section 7.2.1, is one that reached the adversarial objective with low auditor friction. Low auditor friction implies that the per-step observation sequence looks, step by step, close to a sequence that a benign scenario could have produced; the attack's adversarial character lies in where the trajectory ends rather than in how each step appears. The reconstruction-based anomaly score is by construction a per-step magnitude, and an episode-mean aggregation dilutes a localized temporal signature across the entire episode length. Figure 7.3.3 confirms this mechanism directly: the per-step reconstruction-error traces of representative successful-attack episodes stay close to the noise floor for the whole episode, indistinguishable from normal traces, while partial-attack traces exhibit large, sustained excursions in the second half of the episode and blocked-attack traces show sharp early spikes that decay as the attack is stopped.

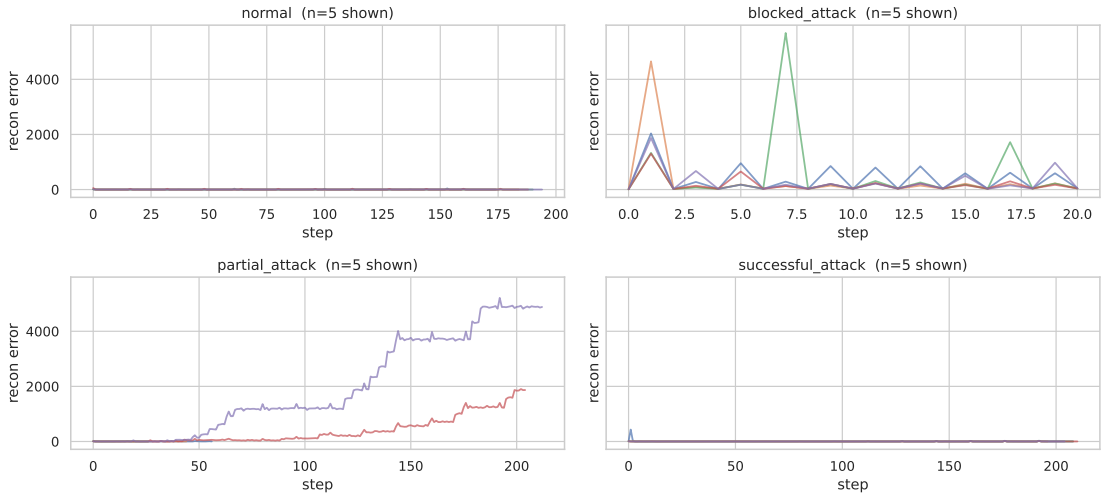

Figure 7.3.3: Per-step reconstruction error traces for five representative episodes from each regime. Normal and successful-attack traces both stay near the noise floor across all steps, producing similar episode-mean ρ. Partial-attack traces show sustained mid-to-late excursions that dominate the episode mean. Blocked-attack traces show sharp early spikes consistent with attack attempts that were interrupted before they could persist. The per-step view makes visible the reconstruction-error signature that the episode-mean summary of Figure 7.3.2 partially hides.

7.3.3 Risk Signal Calibration

The raw reconstruction error ρ is unbounded, heavy-tailed, and not directly comparable across models; downstream consumption by the POMDP observation kernel of Section 7.4 requires a calibrated risk $r = \varphi(\rho) \in [0, 1]$ whose magnitude reflects the posterior probability of anomaly rather than the raw error magnitude. Three calibrators were fit on a held-out calibration split of $n = 66$ normal and $n = 66$ attack episodes, distinct from both the training and the final evaluation splits: a percentile calibrator that maps ρ to its rank against normal-validation errors, a logistic calibrator fit on log-transformed ρ, and a ten-bin isotonic (binning) calibrator. All three were evaluated under 5-fold stratified cross-validation with three metrics: in-sample Expected Calibration Error (ECE), out-of-fold ECE, and out-of-fold ROC-AUC. Results are reported in Table 7.3.1.

Table 7.3.1: Calibrator comparison on the held-out calibration split ($n_{\text{normal}} = 66$, $n_{\text{attack}} = 66$), under 5-fold stratified cross-validation. The honest comparison is on out-of-fold ECE; in-sample ECE is reported only to surface the gap between apparent and true calibration quality.

Calibrator	ECE_{in}	ECE_{oof}	AUC_{oof}	n_{distinct}	Eligible
percentile	0.143	0.133	0.891	60	yes
logistic	0.062	0.089	0.909	66	yes
binning	0.000	0.060	0.902	5	no

The binning calibrator has the lowest out-of-fold ECE (0.060) but is rejected by the granularity gate: with only five distinct output values on the normal-validation risks, the quantile-based severity thresholds collapse (the 50th and 95th percentiles of normal-validation risks fall in the same bin, producing $\tau_2 = \tau_3$ to within rounding).

The granularity floor of 20 distinct values ensures the POMDP observation kernel of Section 7.4 receives a non-degenerate distribution. The logistic calibrator is chosen: second-lowest ECE (0.089), AUC within noise of the best (0.909 vs 0.902), 66 distinct values on the normal-validation risks, and severity bands of width 0.219 and 0.067 that clear the minimum-width guard of 0.05. The three severity thresholds derived from the 50th, 95th, and 99th percentiles of the calibrated normal-validation risks are $\tau_1 = 0.241$, $\tau_2 = 0.460$, $\tau_3 = 0.527$. Figure 7.3.4 reports the corresponding FPR and FNR sweep across the calibrated-risk range, annotated with the three severity thresholds.

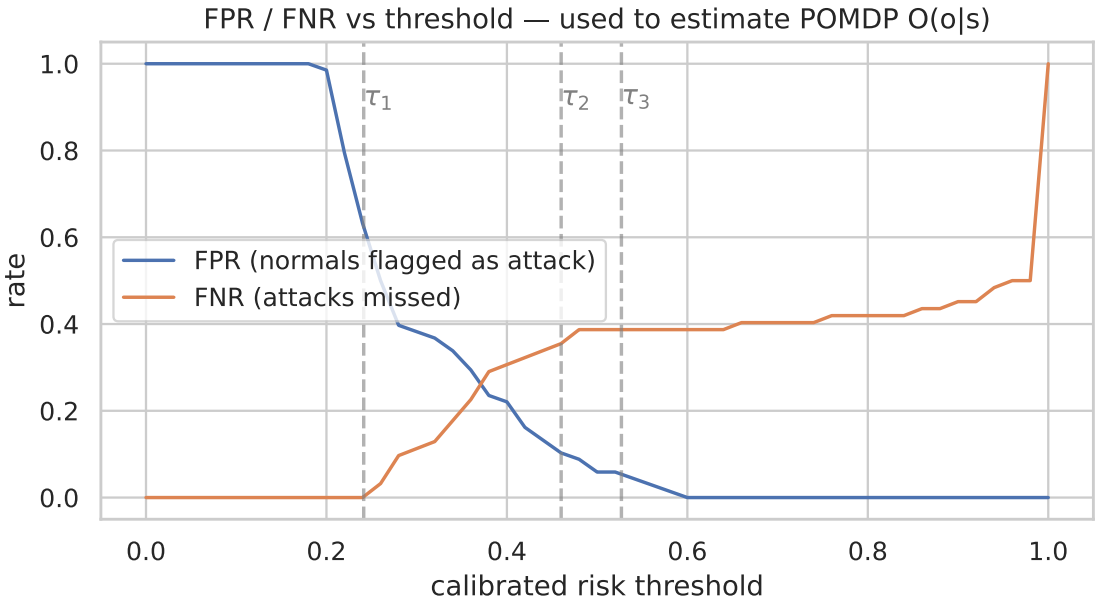

Figure 7.3.4: False-positive rate (FPR, normals flagged as attack) and false-negative rate (FNR, attacks missed) as functions of the calibrated-risk threshold. The three severity thresholds τ_1, τ_2, τ_3 are marked. The curve exhibits a sharp knee around $\tau \approx 0.4$: tightening the threshold from 0.4 to 0.6 reduces FPR from 22% to 0% at the cost of an FNR increase from 31% to 39%, and further tightening above τ_3 produces almost no additional FNR reduction while leaving FPR at zero. No single threshold simultaneously achieves low FPR and low FNR, which is the quantitative statement of the baseline-insufficiency argument in Section 7.1 and the principal motivation for the temporal-belief filter of Section 7.4.

The FN/FP curve of Figure 7.3.4 is reported separately because the threshold grid it sweeps is precisely the input required by the POMDP observation kernel $O(o \mid s)$ of Section 7.4. For each candidate threshold τ the observed emission probabilities $\Pr(o = \text{alert} \mid s = \text{benign}) = \text{FPR}(\tau)$ and $\Pr(o = \text{no-alert} \mid s = \text{compromised}) = \text{FNR}(\tau)$ are tabulated into `fn_fp_table.csv` and consumed verbatim by the belief-update step downstream.

7.3.4 Attack-Class Detection Breakdown

The primary regime, defined in Section 7.2.1 as the union of successful and partial attacks ($n = 113$), is the evaluation regime the thesis claims must be detected: these are the episodes where the static auditor did not fully stop the adversary and dynamic IAM must add value. The GMM-prior CVAE+LSTM achieves ROC-AUC $= 0.888$ on the primary regime, a gain of $+0.243$ over the metadata-only logistic-regression baseline of Section 7.2.2 (AUC $= 0.645$). Precision-recall AUC on the same regime is 0.937.

Table 7.3.2 reports the breakdown across the four regime slices the evaluation protocol uses.

Table 7.3.2: Episode-level detection performance of the GMM-prior CVAE+LSTM, broken down by regime. Primary is the headline number; the successful and partial rows decompose it, and the secondary (blocked) and aggregate rows are reported for completeness only. The metadata baseline is the 5-fold ROC-AUC of a logistic regression on seven episode-level aggregates from Section 7.2.2.

Regime	Role	n	ROC-AUC	PR-AUC
Primary (successful + partial)	Headline	113	0.888	0.937
Successful	Hardest sub-regime	41	0.801	—
Partial	Easier sub-regime	72	0.938	—
Secondary (blocked)	Sanity check	32	0.853	0.762
Aggregate	Reporting only	145	0.880	0.943
Metadata-only baseline	Leakage floor	145	0.645	—

Three observations from Table 7.3.2 govern the interpretation of the remaining subsections in this chapter.

Partial attacks are detected cleanly; successful attacks are not. The 0.138 gap between partial-attack AUC (0.938) and successful-attack AUC (0.801) is, geometrically, the overlap visible in Figure 7.3.2 and, mechanistically, the consequence of the per-step trace behaviour shown in Figure 7.3.3. A successful attack produces a trajectory that the encoder can reconstruct almost as well as a benign trajectory, because successful attacks are precisely those that navigate to the adversarial objective without tripping the auditor, and tripping the auditor is one of the principal mechanisms by which an attack produces a high reconstruction error under the current feature set. At the operating point $\tau \approx 0.5$, the detector recalls 61% of attack episodes at a false-positive rate of 6% and a precision of 0.90; the remaining 39% missed attacks are concentrated in the successful-attack regime. This is a real limitation of the per-step reconstruction-error score as a stand-alone signal and is the quantitative evidence for the argument that point-in-time risk estimation is insufficient for regulating temporally-evolving behaviour (Section 3.4).

Blocked attacks are detected better than successful attacks. The 0.853 AUC on blocked attacks is higher than the 0.801 AUC on successful ones, which at first reading is counter-intuitive: blocked attacks are, by construction, the adversarial episodes that the auditor stopped, and one might expect them to be the easiest to flag. The mechanism is the inverse of the successful-attack one: a blocked attack typically produces a short, aborted sequence of rejected actions, and those rejected actions generate sharp early spikes in the reconstruction error (visible in the blocked panel of Figure 7.3.3) that survive the episode-mean aggregation because the episode itself is short. The practical consequence is that the reconstruction-error signal is most discriminative on the attack class where dynamic IAM adds the least value (the auditor already stopped the attack) and least discriminative on the class where it must add the most value (successful attacks). Resolving this inversion is the task of the historical-trust layer of Section 7.4.

Gains over the metadata baseline are substantial across all regimes. The primary-regime gain of $+0.243$ over the logistic baseline (from 0.645 to 0.888) exceeds the

gain attributable to any single high-MI feature in Section 7.2.2 and is larger than the spread across calibrator variants in Table 7.3.1. This rules out the hypothesis that the CVAE+LSTM's discriminative performance is a smoothed restatement of the metadata summaries available to the baseline. The quantitative support for this claim is extended by the ablation of Section 7.3.7, which shows that a substantially larger fraction of the gap is carried by features that are not visit-intensity aggregates.

7.3.5 Latent-Space Geometry

The 16-dimensional latent space $\mu(x)$ is the representation the downstream trust-POMDP consumes as context for the belief update. Its regime-discriminative geometry is evaluated along two axes: a silhouette score on the full 16-dimensional latent, and a UMAP projection to two dimensions for visual inspection. The silhouette score of regime labels in the 16-dimensional latent is 0.366, a modest but positive value that confirms the regimes occupy distinguishable regions of the latent without clean partitioning. This is the expected outcome for a CVAE trained on normal data only: the encoder's objective is to reconstruct normal trajectories, not to classify regimes, so clean regime separation in the latent is not a training target but a downstream benefit that accrues to the extent that attack trajectories fall off the normal manifold. Figure 7.3.5 visualizes the latent structure. The left panel colours all four regimes with normals rendered as a grey background layer (because the 444:145 class imbalance otherwise washes out the attack structure); the right panel greys out normals entirely and separates the three attack regimes by colour. The visible geometry matches the detection numbers of Table 7.3.2: successful attacks (blue) anchor the right-hand spur of the embedding, partial attacks (green) dominate the left-hand arm, and blocked attacks (pink) concentrate at the top of the left arm. Crucially, the successful-attack cluster extends into the normal grey cloud more than the partial cluster does, which is the geometric form of the same observation made in the reconstruction-error view: successful attacks are locally similar to normal trajectories in the representation the encoder learned.

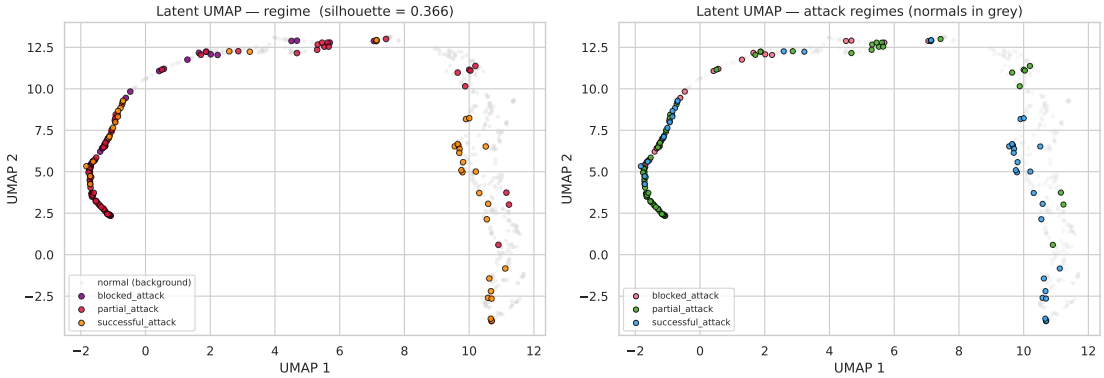

Figure 7.3.5: UMAP projection of the 16-dimensional per-episode latent means. Left: all four regimes, with normals rendered as a grey background layer to compensate for the $3:1$ class imbalance. Right: attacks only, with normals greyed out entirely. The three attack regimes occupy distinguishable but overlapping regions of the embedding; successful attacks encroach most heavily into the normal cloud, partial attacks concentrate on a distinct arm, and blocked attacks form a compact cluster at the top-left. Silhouette of regime labels in the 16-dimensional latent: 0.366.

7.3.6 Ablation: Prior Choice

The Best model selection rule used in the training process uses the rule of ”pick on primary-regime AUC”; based on this selection choice the GMM-prior variant by a margin of 0.002 (0.888 vs 0.886) was selected. Both priors produce AUCs within 0.01 of each other on every regime slice, and the GMM variant's apparent advantage on the hardest sub-regime (successful-attack AUC 0.801 vs 0.775) is on a base of 41 episodes, where a single episode reclassification moves the rate by more than two percentage points. The honest statement is that the two priors are within the statistical resolution this dataset supports. The GMM variant is reported as the champion in the interest of internal consistency with the pre-defined selection rule, and the standard-Normal variant is retained as the reference for the visit-count ablation in Section 7.3.7 for a distinct reason (the ablation was designed and debugged against the standard-Normal run). No claim is made that the additional mixture parameters of the GMM prior are discriminatively useful on this dataset.

7.3.7 Ablation: Visit-Count Features

The nine per-node visit counters `visits_Public_Search` through `visits_Production_DB` carry a strong intensity asymmetry between regimes, documented in Section 7.2.2: Zone 3 nodes are visited 40–120 times more often by primary-regime attack episodes than by normal episodes. This asymmetry is a legitimate behavioural signal under the threat model, but its magnitude is a direct consequence of the attack portfolio used in this experiment, and a learned detector could in principle derive most of its discriminative performance from the visit-count block alone rather than from the sequential structure the LSTM is meant to capture. To bound this contribution, the champion model was retrained with the nine visit-count columns zeroed out at the input level, leaving a 12-dimensional raw-observation block plus the unchanged numeric-derived and categorical blocks. Training duration and all other hyperparameters were held constant. Results are reported in Table 7.3.3.

Table 7.3.3: Feature-group ablation on the nine visit-count columns. Both models are trained for 300 epochs on the same episode-level training/validation/test split. The ablated model's primary-regime AUC is within noise of the full model's; the small positive delta on secondary-regime AUC is consistent with the visit-count block carrying some attack-class-specific noise that the abridged input set does not propagate.

Feature set	**Primary AUC**	**Secondary AUC**	**Aggregate AUC**
Full (21 dims)	0.888	0.853	0.880
Without visit counts (12)	0.882	0.954	0.898
Δ	-0.006	$+0.101$	$+0.018$
Metadata-only floor		0.645	

The ablated model's primary-regime AUC of 0.882 is within 0.006 of the full model's 0.888, a difference smaller than the spread between prior variants in Section 7.3.6 and well within the resolution supported by the $n = 113$ primary-regime sample size. The detector therefore does not derive its discriminative performance from the visit-count block; it derives it from the guard counters, the action and auditor-decision streams,

the resource-sensitivity channel, and the temporal structure the LSTM captures across the $W = 32$-step receptive field. This is the strongest of the four outcomes enumerated in the pre-registration discussion of Section 7.2.2: the detector is essentially ignoring the visit counts at the representation level. The counter-intuitive secondary-regime gain ($+0.101$) is not attributable to any single factor; the most plausible explanation is that the visit-count block carries some attack-class-dependent noise on blocked attacks (whose Zone 3 access pattern is interrupted mid-attack) that the ablated model is not burdened with, but the magnitude of the shift is within the $n = 32$ binomial resolution on the blocked regime and is reported without interpretive claim.
The practical consequence for external validity is substantial: the detector's performance on primary-regime attacks does not depend on the visit-intensity asymmetry that is itself a function of how the attack library was written.

7.3.8 Interpretation — Feature Attribution

Integrated Gradients [84] was applied to the champion model to attribute per-window reconstruction error to individual input features. The attribution was computed against a zero-valued baseline over 32-step windows sampled from each regime, with cuDNN disabled to permit backward-pass evaluation of the LSTM in `eval` mode. The absolute attribution sums per feature over a full window are approximately balanced across regimes ($\sum |\text{IG}|$ ranges from 102.6 on blocked-attack windows to 112.8 on normal windows), indicating that the total gradient magnitude is a property of the encoder rather than of the regime, and that regime differences live in the distribution of attribution across features rather than in its total mass. Qualitative inspection of the per-feature attribution profiles shows that the guard counters (`rejection_count`, `no_progress_count`, `consecutive_denied`) and the `auditor_decision` categorical dominate attribution on partial and blocked attacks, while successful-attack attribution is more diffusely spread across the full feature set with no single dominant driver. This is consistent with both the reconstruction-error signature of Figure 7.3.3 and the ablation result of Section 7.3.7: the features that carry the partial/blocked signal are behavioural friction indicators, not node-identity channels, and successful attacks are hard to detect precisely because they do not excite these friction indicators.

7.4 Historical Trust Estimation — RQ2

Section 7.3 established that the CVAE+LSTM stage produces a calibrated instantaneous risk $r_t \in [0, 1]$ that is informative but noisy, and that its primary weakness is under-detection of successful attacks relative to partial and blocked attacks. That weakness is a property of any instantaneous-risk signal: successful attacks concentrate their adversarial signature in a small window of steps, so the episode-mean risk dilutes the evidence below the threshold of a single-point classifier. The POMDP of Section 7.4 was designed to address this weakness by integrating evidence over time: the belief state b_t is a Bayes-filtered posterior over the four hidden states $\mathcal{S} = \{\text{safe}, \text{degraded}, \text{danger}, \text{deadend}\}$ and trust $\tau_t = \sum_s w_s b_t(s)$ is a weighted linear functional of that belief. This section empirically characterises the filter on the evaluation dataset and measures its contribution to episode-level attack detection.

7.4.1 Evaluation Protocol and Artefacts

The POMDP is constructed from three persisted artefacts: the thresholds $(\tau_1, \tau_2, \tau_3) = (0.241, 0.460, 0.527)$ obtained from the logistic calibrator of Section 7.3, the empirical false-positive/false-negative sweep, and the trust-weight vector $w = (1.0, 0.67, 0.33, 0.0)$ for states $(\text{safe}, \text{degraded}, \text{danger}, \text{deadend})$. The observation kernel $O(o \mid s)$ is constructed by reading the SAFE emission row directly from the FPR sweep on normal episodes and the DEADEND emission row from the attack-class recall $1 - \text{FNR}$, and interpolating the two interior rows as a monotone blend:

$$P(o \mid \text{degraded}) = 0.75\, P(o \mid \text{safe}) + 0.25\, P(o \mid \text{deadend}), \tag{7.4.1}$$

$$P(o \mid \text{danger}) = 0.25\, P(o \mid \text{safe}) + 0.75\, P(o \mid \text{deadend}). \tag{7.4.2}$$

This is the smallest assumption consistent with the qualitative ordering "degraded is closer to safe, danger is closer to deadend" and produces a monotonically non-decreasing $P(\text{critical} \mid s)$ across the four hidden states, which is the structural property the filter requires to monotonically decrease trust under sustained alerts. Laplace smoothing $\alpha = 10^{-2}$ is applied to every emission cell to prevent catastrophic belief collapse on a single mis-detection. The transition matrix $T(s' \mid s)$ is handcrafted to encode the slow-degradation prior from Section 7.4: the SAFE state is sticky (0.97), DEADEND is near-absorbing (0.97), and the recovery-to-progression asymmetry from DANGER is 0.05 versus 0.30. The initial belief is $b_0 = (0.95, 0.04, 0.01, 0.00)$. Both kernels are visualised in Figure 7.4.1.

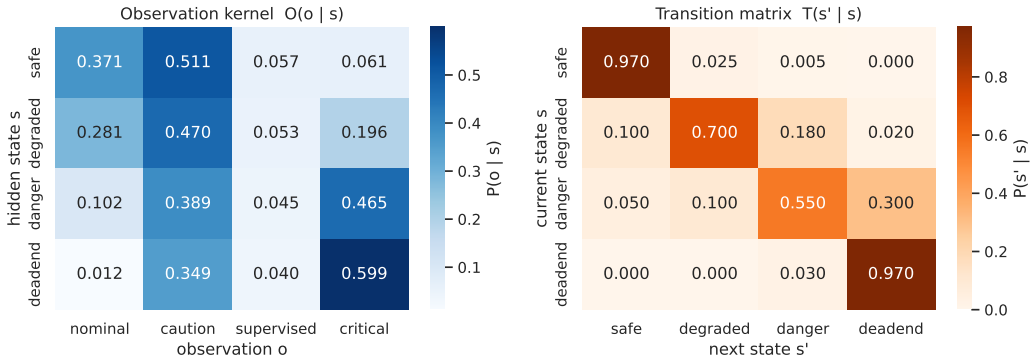

Figure 7.4.1: POMDP observation kernel $O(o \mid s)$ (left) and transition matrix $T(s' \mid s)$ (right). The observation kernel satisfies the monotonicity property $P(\text{critical} \mid s)$ non-decreasing in severity; the transition matrix encodes the slow-degradation asymmetry.

All 589 episodes are evaluated. For each episode, the CVAE+LSTM is applied with stride 1 over a sliding window of $W = 32$ steps to produce a per-step calibrated risk r_t; the filter is reset to b_0 at the start of each episode and the Bayes update of Section 7.4 is applied at every valid step. The resulting trust trajectory $\tau_{1:T}$ and belief vectors $b_{1:T} \in \Delta^3$ are summarised into a six-dimensional per-episode feature vector:

- $\bar{\tau}$ — mean trust over valid steps;
- τ_{final} — trust at the last valid step;
- τ_{min} — minimum trust over valid steps;

- $\text{frac}_{\text{unsafe}}$ — fraction of steps with $\tau < 0.5$;
- $\text{frac}_{\text{deadend}}$ — fraction of steps whose MAP hidden-state estimate is `deadend`;
- $\bar{r}$ — mean calibrated risk (the CVAE-only baseline).

The sixth feature $\bar{r}$ is retained to enable direct computation of the POMDP's marginal contribution over the RQ1 baseline.

7.4.2 Belief Dynamics on Benign Trajectories

Before conducting the real experiment,three sanity trajectories designed to verify that if the filter behaves as intended. Figure 7.4.2 shows the trust trajectory under (A) fifty steps of normal observations ($r = 0$), (B) fifty steps of sustained critical observations ($r = 1$), and (C) a burst of ten critical observations embedded in a normal context.

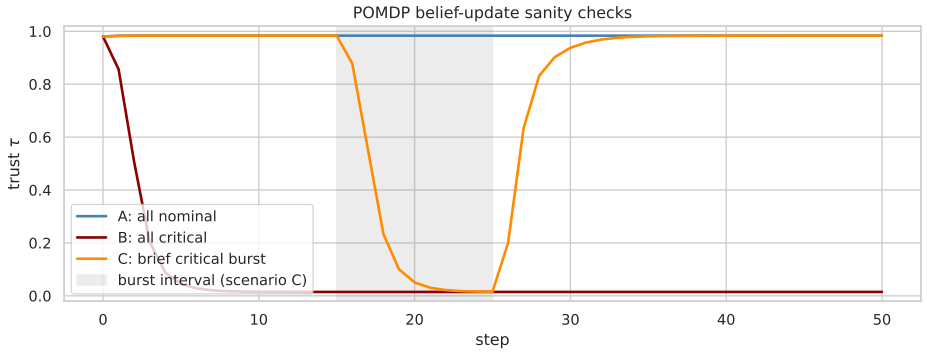

Figure 7.4.2: Belief-update sanity trajectories. Under sustained normal observations the filter remains at $\tau \approx 0.98$; under sustained critical observations it collapses monotonically to $\tau \approx 0.015$; under a brief critical burst surrounded by nominal context the trust dips and then recovers. The hysteresis visible in scenario C is the property that lets the filter be robust to sensor noise while still being sensitive to sustained anomalies.

All three trajectories behave as specified. Also, further evaluation on the normal dataset shows that on the 444 benign episodes of the evaluation set the POMDP produces a tightly concentrated mean-trust distribution around $\bar{\tau} \approx 0.81$ (standard deviation 0.03), confirming that the filter does not introduce genuine drift on clean trajectories. We believe, the slight offset below 1.0 reflects the finite FPR of the underlying detector: a small fraction of legitimate steps produce risk values above τ_1, and the filter correctly registers that evidence, without letting it accumulate. (One reason for this event is the fact that due to the limitations of experiment, in some normal scenarios the auditor reject the requests of the agent in the first few steps and this creates a spike in risk signal at the beginning but vanishes over horizon)

7.4.3 Belief Response to Attacks

The filter's behaviour on attack episodes is characterised at two levels of aggregation. Figure 7.4.3 shows one representative trajectory per regime, selected to illustrate the dynamics each regime is capable of exhibiting; Figure 7.4.4 shows the median trajectory and inter-quartile band over all episodes of each regime, which is the population-level evidence for RQ2.

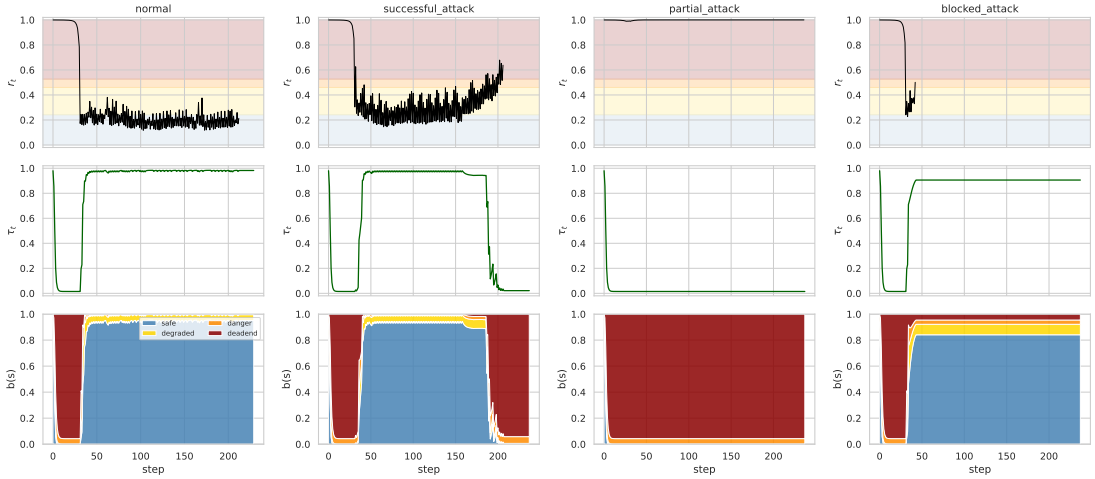

Figure 7.4.3: Representative risk, trust, and belief trajectories, one per regime. Each column shows calibrated risk r_t (top), filtered trust τ_t (middle), and the belief vector b_t as a stacked area plot (bottom). Exemplars are picked by a regime-specific scoring rule that rewards the qualitative behaviour the section claims to demonstrate, rather than by episode length, which would systematically select degenerate cases in which r_t is saturated near 1 from the first step.

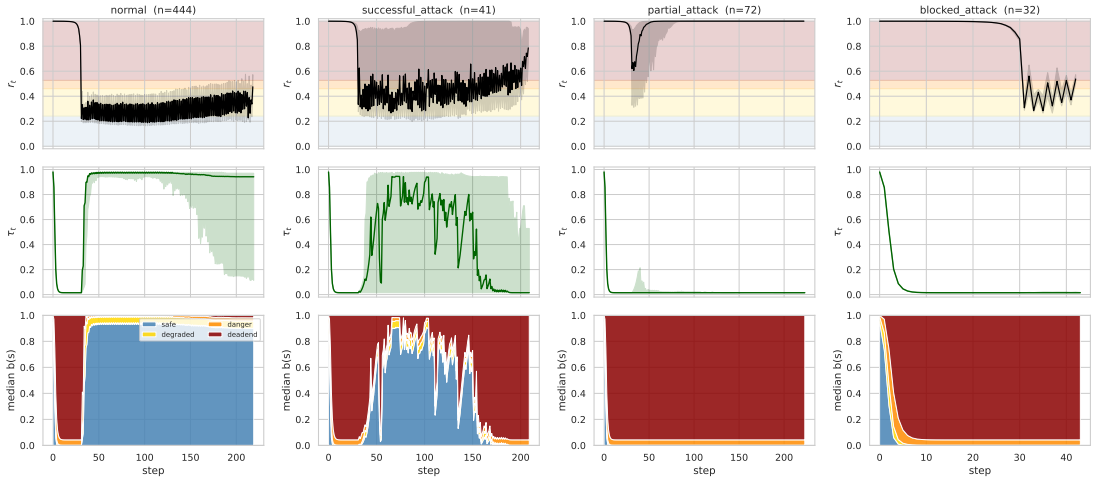

Figure 7.4.4: Median trajectories and 25–75% inter-quartile band over all episodes of each regime. The horizon is truncated to the longest step at which at least eight episodes still contribute. Episodes are aligned at step 0, which is the alignment available at deployment time.

On normal episodes the median trust climbs to $\tau \approx 0.95$ within roughly forty steps and remains there. (a small fraction of legitimate startup observations cross τ_1 and produce a brief excursion toward deadend that the filter then correct it.)

On successful-attack episodes the median per-step risk lies in the caution–supervised band ($r_t \in [0.3, 0.6]$) for most of the episode and crosses into the critical band only in narrow windows. This is precisely the regime where the CVAE mean dilutes the signal and where the temporal filter is supposed to help. The aggregate trust trajectory shows this directly: τ collapses on the leading critical window, recovers partially, and

then collapses durably as further critical observations accumulate. The final-step trust is near zero on the median (Which also converts the adversarial signature into a usable per-episode summary).

On partial-attack episodes the median r_t is sustained near 1.0 throughout, and the filter saturates on deadend within a handful of steps. The recovery dynamics the prior intuition anticipated for this regime do not characterise the median; they appear only on individual episodes and are not a reliable population-level property. This is consistent with the per-regime AUC of Section 7.4.4, where partial attacks are the easiest of the three regimes for both the CVAE and the POMDP-augmented score.

Blocked-attack episodes are short by construction (the auditor truncates the adversarial phase early) so the effective horizon is narrow and the filter completes its excursion in fewer than fifty steps. The exemplar shows the brief dip and recovery the regime is named for; the aggregate shows that on the median the dip is sharper and the recovery does not fully complete before the episode ends.

7.4.4 Discriminative Power of Trust-Derived Features

The central empirical question for RQ2 is whether the trust-derived features separate the two classes (normal vs. attack) better than the CVAE baseline $\bar{r}$. Table 7.4.1 reports the threshold-free ranking metrics for all six per-episode features.

Table 7.4.1: Per-feature threshold-free ranking metrics (ROC-AUC and PR-AUC) on the full normal-vs-attack task. Features derived from the POMDP are grouped at the top; the CVAE baseline $\bar{r}$ is at the bottom for comparison.

Feature	ROC-AUC	PR-AUC
$\bar{\tau}$	0.890	0.780
τ_{final}	0.887	0.766
$\text{frac}_{\text{deadend}}$	0.882	0.744
$\text{frac}_{\text{unsafe}}$	0.882	0.744
τ_{min}	0.686	0.582
$\bar{r}$ (CVAE)	0.876	0.698

Four of the five POMDP-derived features beat the CVAE baseline on both AUCs. The mean trust $\bar{\tau}$ achieves the highest ROC-AUC (0.890) and PR-AUC (0.780), an improvement of $+0.014$ and $+0.082$ over $\bar{r}$. The feature τ_{min} is an outlier (it under performs because the filter occasionally reaches near-zero trust) briefly on normal episodes when a single noisy step produces a critical observation, and the minimum is not robust to such single- step excursions. This reinforces the methodological argument of Section 7.4 that trust is a filtered, history-aware quantity rather than a point statistic.

At the operating point that maximises balanced accuracy, τ_{final} accuracy is 0.817 at a threshold of 0.054, compared with 0.817 at threshold 0.658 for $\bar{r}$. The two are statistically indistinguishable at this operating point, but their The confusion matrices of trust and risk differ systematically: τ_{final} captures 122 of 145 attacks (recall 0.841) at the cost

of 92 false positives on normals, whereas $\bar{r}$ captures 118 attacks (recall 0.814) with 80 false positives. The trust is preferring recall, while the risk is preferring precision. Neither ordering is inherently preferable, which motivate us to ask the relevant question in which attack subclass each variant is better at catching. We try to answer this question in the next part.

7.4.5 Per-Regime Breakdown — Successful-Attack Recovery

The main claim of RQ1 was that the CVAE+LSTM under-detects successful attacks relative to partial and blocked attacks. If the POMDP's filtering is doing useful work, the clearest place to see that work is in the successful-attack regime. Table 7.4.2 decomposes the ROC-AUC of three scoring functions against each attack regime separately, using the normal split as the shared negative class.

Table 7.4.2: Per-regime ROC-AUC (normals vs. each attack regime) for three scoring functions. Successful attacks are the RQ1 bottleneck; the POMDP single-feature scorer closes most of the gap to the partial-attack rate.

Score	**PRIMARY** (succ+partial)	**successful** attack	**partial** attack	**blocked** attack
CVAE ($\bar{r}$)	0.857	**0.751**	0.918	0.943
POMDP (τ_{final})	0.885	**0.805**	0.931	0.892
CVAE + POMDP joint (5-fold CV)	0.872	0.742	0.947	0.941

The POMDP single-feature scorer τ_{final} improves ROC-AUC from 0.751 to 0.805, a gain of $+0.054$, while preserving the partial-attack AUC at 0.931 (within $+0.013$ of the CVAE baseline). The joint CVAE–POMDP scorer does not improve successful-attack discrimination over final-step trust alone. Consequently, the controller evaluation retains instantaneous risk and filtered trust as separate, interpretable inputs rather than claiming an empirically optimal statistical fusion. Developing a principled fusion mechanism is left for future work.

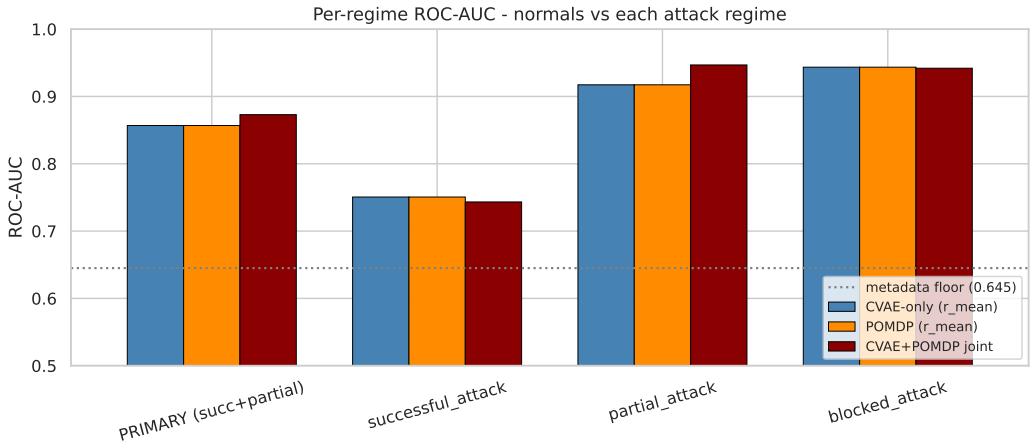

Figure 7.4.5: Per-regime ROC-AUC (normals vs. each attack regime). The partial and blocked regimes are near-saturated for all scorers; the successful-attack regime is where the POMDP's temporal integration contributes most. The dotted horizontal line at 0.645 marks the metadata-only floor reported in Section 7.3.

7.4.6 Summary

The POMDP of Section 7.4 converts the noisy instantaneous risk produced by the CVAE+LSTM into a filtered, history-aware trust scalar τ whose per-episode summary features ($\bar{\tau}$, τ_{final}, $\text{frac}_{\text{deadend}}$) improve on the CVAE-only baseline both in aggregate ROC-AUC (+0.014 on the full task) and, more importantly, on the successful- attack subregime identified as the RQ1 bottleneck (+0.054). The filter's sanity behaviour on benign and adversarial trajectories matches the theoretical specification: it is stable under nominal observations while collapsing under sustained critical observations. The per-episode summary artefacts produced in this section are the input to the MPC-based access controller evaluated in Section 8.1.3.

7.5 Adaptive Access-Control Evaluation under Offline Replay (RQ3)

The first two research questions produce, at every step t, a calibrated risk $r_t \in [0, 1]$ (RQ1) and a filtered trust scalar $\tau_t \in [0, 1]$ (RQ2). RQ3 asks how to act on these signals: given (r_t, τ_t) and the POMDP belief b_t, which level of autonomy should the agent be granted? We cast this as a control problem and compare three controllers that assign one of four ordered autonomy roles — full, supervised, prescribed, quarantined, with integer levels $L \in \{0, 1, 2, 3\}$ — to each agent at each step. The three controllers are a soft-threshold policy (a linear rule), a discrete finite-horizon quadratic-cost controller, and a barrier-inspired threshold safety filter.

All experiments in this chapter run on the replay of pre-recorded episodes: 444 benign and 145 attack episodes (split into 41 successful, 72 partial, and 32 blocked attacks). Because r_t, τ_t and b_t are produced upstream, every controller sees identical inputs and the comparison isolates the decision rule. The implications of this offline setup are discussed in Section 8.2.

7.5.1 Evaluation Protocol

Action granularity and ground truth. The evaluation dataset provides episode-level labels (normal, successful attack, partial attack, blocked attack) but no step-level annotation of which individual steps constitute adversarial activity. An operational surrogate is used in their place: at every step the tensor contains a categorical `action_type_id` identifying which action the agent invoked (navigate, invoke_tool, escalate, observe, inspector, or the no-tool-call sentinel). Using the action-criticality mapping defined in Appendix A, Equation (A.7.44), a step is considered a *sensitive action opportunity* only if *both* the action is capable of state change (weight $w_a \geq 0.35$) *and* the agent is at a sensitive node. Node sensitivity is read directly from the standardised `resource_sensitivity` channel of the feature tensor (column index 1 of the raw-observation block), un-standardised through the scaler stored in the preprocessing manifest. A node is considered sensitive if resource_sensitivity ≥ 0.3; this threshold corresponds to the three critical-asset nodes of the grid (`Admin_Panel`, `Production_DB`, and `Financial_Gateway`) and is justified empirically by the threshold-sweep reported below.

The two-factor definition matters quantitatively. Of the 8,047 `invoke_tool` invocations on normal episodes, only 30% occur at high-sensitivity nodes; the corresponding rate on attack episodes is 66%. Treating all `invoke_tool` calls as equally sensitive would therefore penalise the controller for blocking legitimate reads on public nodes, which is a property of the evaluation definition rather than the controller itself. The node-aware definition is documented in the evaluation notebook and is used for every result in this section.

Offline-replay metrics. The controller's decisions define two competing key performance indicators on the sensitive-action steps identified above:

- **Harm-prevention rate (HPR)** — on attack episodes, the fraction of sensitive-action attempts that the assigned role disallows:

$$\text{HPR} = \frac{\#\{\text{sensitive attacks blocked}\}}{\#\{\text{sensitive attacks attempted}\}}.$$

  Higher is better. Because the dataset does not contain step-level malicious-action annotations, HPR is an operational proxy rather than a direct measurement of prevented harm. It measures the fraction of sensitive action opportunities in attack-labelled episodes that the replayed policy denies.

- **Benign-action rate (BAR)** — on normal episodes, the fraction of sensitive-action attempts that the assigned role *permits*:

$$\text{BAR} = 1 - \frac{\#\{\text{sensitive benign actions blocked}\}}{\#\{\text{sensitive benign actions attempted}\}}.$$

  Higher is better.

These two quantities are fundamentally in tension: a controller that quarantines every agent achieves HPR $= 1$ at BAR $= 0$; a controller that grants every action achieves BAR $= 1$ at HPR $= 0$. The controller's value is characterised by the trade-off curve it traces on the (BAR, HPR) plane as the decision thresholds are varied.

7.6 Three controller formulations

Shared suspicion signal. All controllers that use the learned signals do so through a single scalar *suspicion* functional,

$$s_t \;=\; \beta_\rho\, r_t \;+\; \beta_\tau\,(1-\tau_t) \;+\; \beta_b\, b_t(\text{deadend}), \tag{7.6.1}$$

with weights $\beta = (\beta_\rho, \beta_\tau, \beta_b) = (2.0,\ 1.5,\ 0.0)$ in our configuration. The term $(1-\tau_t)$ converts trust into distrust so that all contributions point in the same direction (higher s_t = more suspicious); the deadend-belief term is disabled here, so $s_t = 2\,r_t + 1.5\,(1-\tau_t) \in [0, 3.5]$.

(i) Soft-threshold policy (linear). The simplest controller places three monotone thresholds $\theta = (\theta_1, \theta_2, \theta_3) = (1.5, 2.5, 3.5)$ on s_t:

$$\text{role}(s_t) = \begin{cases} \text{full} & s_t < \theta_1, \\ \text{supervised} & \theta_1 \le s_t < \theta_2, \\ \text{prescribed} & \theta_2 \le s_t < \theta_3, \\ \text{quarantined} & s_t \ge \theta_3. \end{cases} \tag{7.6.2}$$

A re-decided role is held for a horizon of $H = 5$ steps before re-evaluation, which acts as hysteresis to suppress role oscillation. Each decision costs three comparisons, i.e. $O(1)$.

(ii) discrete finite-horizon quadratic-cost controller. The general formulation treats the role assignment as a receding-horizon optimization over a sequence of role one-hot vectors $U = (u_t, \dots, u_{t+H-1})$, $u_k \in \{0,1\}^4$, $\mathbf{1}^\top u_k = 1$. With $L_j \in \{0,1,2,3\}$ the level of role j, the objective is

$$\min_{U} \sum_{k=0}^{H-1} g(\hat{s}_{t+k})^\top u_{t+k} \;+\; \lambda_{\text{s}} \sum_{k=-1}^{H-2} \left\| L\,u_{t+k+1} - L\,u_{t+k} \right\|^2, \tag{7.6.3}$$

where u_{t-1} is the currently held role and the per-role stage cost is *convex* (quadratic) in the level,

$$g_j(\hat{s}) = \lambda_{\text{risk}}\, \hat{s}\, \left(\tfrac{3-L_j}{3}\right)^2 \;+\; \lambda_{\text{util}} \left(\tfrac{L_j}{3}\right)^2. \tag{7.6.4}$$

The first term (*exposure*) makes permissive roles expensive when suspicion is high; the second (*friction*) makes restriction always cost something, so the optimum is an interior role that slides from full to quarantined as $\hat{s}$ grows. The switching term $\lambda_{\text{s}} \| \cdot \|^2$ penalises changing autonomy level between adjacent steps. Predicted suspicion $\hat{s}_{t+k}$ is produced by a simple autonomous forward model that mean-reverts the driving signals, $\hat{r}_{t+k} = r_t \gamma^k + r_{\text{base}}(1-\gamma^k)$ (and likewise for $1-\tau$), with $\gamma = 0.85$; without such a model the H-step plan would be constant and the program would collapse to the single-step minimiser. We use $\lambda_{\text{s}} = 0.5$ and calibrate λ_{util} as the controller's strictness knob. The discrete optimization problem is solved exactly by enumerating the 4^H candidate sequences, which is $O(4^H)$ per re-optimization.

(iii) Barrier-inspired threshold safety filter. The third controller discards the optimizer and forces the role purely from three hard thresholds on the instantaneous signals, defaulting to full when none fires:

$$\begin{aligned} \tau_t < 0.30 &\Rightarrow \text{prescribed}, \\ r_t > 0.50 &\Rightarrow \text{prescribed}, \\ r_t > 0.95 &\Rightarrow \text{quarantined}. \end{aligned} \tag{7.6.5}$$

The strictest triggered threshold wins, applied immediately with no horizon hold; it is memoryless and reacts only to the present step.

7.7 Linear versus Finite-Horizon Controller

The quadratic reduces to the linear rule on a discrete action space. With four ordered roles, the continuous minimiser of (7.6.4) at $H = 1$ is $L^\star(\hat{s}) = 3\lambda_{\text{risk}}\hat{s}/(\lambda_{\text{util}} + \lambda_{\text{risk}}\hat{s})$, which increases monotonically with suspicion. Rounding a monotone scalar to the nearest of four levels is exactly equivalent to placing three thresholds on $\hat{s}$, i.e. the soft-threshold rule (7.6.2). We verify this empirically: sweeping s from 0 to 3.5, the QP at $H = 1$ walks through full→supervised →prescribed→quarantined as s crosses fixed breakpoints. The quadratic curvature only matters if it changes which of the four roles is optimal, and on this simplified, low-resolution grid of allowed states (where values are separated by significant, discrete steps) it generically does not (the rounding quantises the curvature away).

Matched-utility comparison. A subtlety is that the QP's friction weight λ_{util} must be calibrated to the data: (7.6.4) grants full only when $\hat{s} < \lambda_{\text{util}}/(5\lambda_{\text{risk}})$, and benign suspicion here has median $s \approx 0.75$, so a small λ_{util} restricts every benign step (this is similar to choosing the right parameters for policy engine that should be done by the access control engineer with an enterprise). We therefore calibrate $\lambda_{\text{util}} = 5.33$ so the QP matches the soft policy's benign-action rate (BAR), and compare at that operating point (Table 7.7.1). The two are statistically indistinguishable: the harm-prevention rate (HPR) differs by one percentage point in favour of the linear rule, well inside the noise scale used elsewhere in this thesis. Comparing whole Pareto frontiers (Figure 7.7.1) confirms this — the mean HPR difference over the shared operating region is -0.0038, i.e. no measurable benefit; while the QP costs ~ 0.16 ms per re-optimization against the linear rule's $\sim 10^{-4}$ ms, a factor of roughly 10^3.

Table 7.7.1: Matched-utility comparison of the linear and quadratic MPC (safety filters off). HPR = harm-prevention rate, BAR = benign-action rate.

Controller	λ_{util}	HPR	BAR	solve time (ms)
Soft-threshold (linear)	—	0.935	0.748	$\sim 10^{-4}$
QP-MPC (quadratic)	5.33	0.925	0.745	0.158

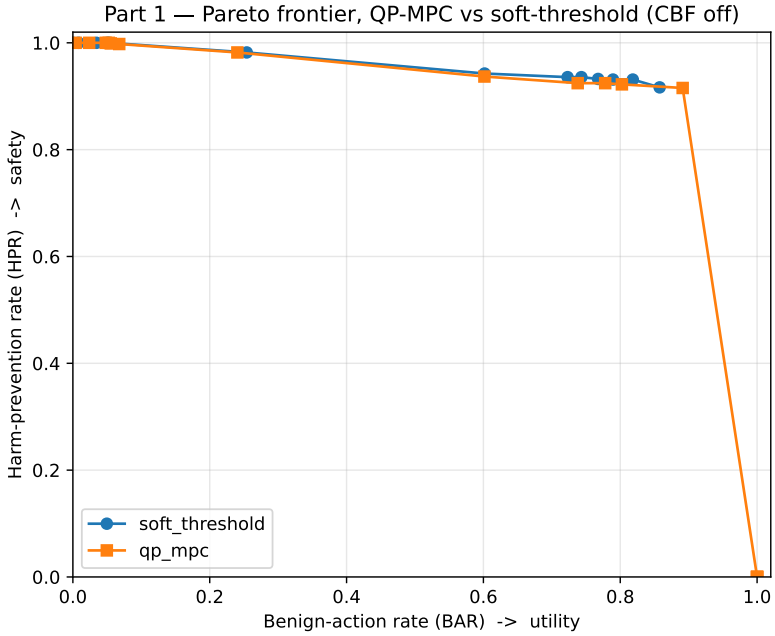

Figure 7.7.1: Safety/utility Pareto frontiers of the linear soft-threshold policy and the quadratic MPC, with barriers disabled. The frontiers overlap within noise; the quadratic formulation buys no measurable advantage at the four-role granularity.

Advantages of the quadratic formulation The equivalence above is specific to our setting: a single agent, a discrete and small action space, and an autonomous forward model. The discrete finite-horizon optimization becomes genuinely necessary once any of these assumptions is relaxed, which is the realistic situation in a multi-agent deployment. With several interacting agents, the natural objective couples their decisions through a non-diagonal Hessian (e.g. limiting the number of agents simultaneously holding high autonomy, or penalising correlated escalations that together exceed a resource budget). Such cross-agent coupling cannot be expressed by independent per-agent thresholds; it requires the joint quadratic form (7.6.3). Similarly, a continuous or finer-grained autonomy variable, or a learned multi-step dynamics model, would make the curvature of (7.6.4) and the horizon meaningful rather than quantised away. We therefore present the quadratic MPC as the general formulation and the linear rule as its exact reduction under the present (single-agent, discrete) action space — deployed because it is identical in decision quality, far cheaper, auditable ("$s_t \geq \theta_3 \Rightarrow$ quarantine" is a one-line justification), and directly tunable by an operator.

Limitations of the quadratic formulation Beyond the $O(4^H)$ cost, the QP introduces three weights ($\lambda_{\text{risk}}, \lambda_{\text{util}}, \lambda_{\text{s}}$) and a forward-model rate γ that have no operator-facing meaning and must be tuned; a mis-set λ_{util} during our experiment silently drove BAR to zero in our runs before calibration. Its behaviour also depends on a dynamics model that an adversary could in principle exploit. The linear rule has none of these failure modes and fails more visibly.

7.8 Barrier-Inspired Safety-Filter Comparison

We next compare the soft-threshold policy with the barrier-inspired threshold safety filter. At the selected default operating points reported in Table 7.8.1, the safety filter is marginally ahead on HPR (0.936 vs 0.935) and reaches full quarantine on 48.8% of successful attacks (median step 37), whereas the soft policy as configured never quarantines (its $\theta_3 = 3.5$ is essentially unreachable by score alone). Comparing frontiers, the safety filter leads by only 0.011 HPR at matched BAR. By the pooled metrics, then, the bare safety filter captures essentially all of the achievable safety/utility trade-off in this environment.

Table 7.8.1: Soft-threshold policy versus barrier-inspired filter at a selected default operating-point comparison. TTQ = median time-to-quarantine on successful attacks.

Controller	HPR	BAR	restriction	TTQ	quar. rate
Soft-threshold	0.935	0.748	0.175	—	0.00
barrier-inspired filter	0.936	0.800	0.151	37	0.49

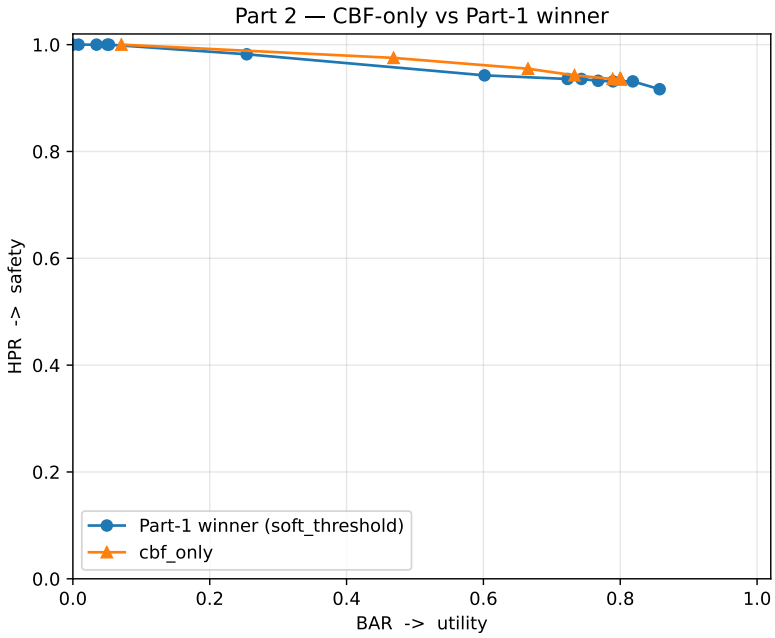

Figure 7.8.1: barrier-inspired filter frontier against the Part-1 winner. The two are close; the safety filter alone recovers most of the trade-off the optimizer achieves.

Limitations of pooled metrics HPR and BAR are evaluated only at sensitive-action steps and ask only "block or allow" questions. Two controllers can produce nearly identical block/allow decisions while behaving very differently over the rest of the episode — and that difference is the point of the layered design. The aggregate offline-replay trace (Figure 7.8.2) makes the divergence visible: on attacks, trust collapses and belief mass moves to the *deadend* state, and the safety filter sits at a

systematically higher mean autonomy restriction than the soft policy (e.g. mean role level 2.90 vs 1.98 on partial attacks). The controllers are not the same rule landing in the same place; they trade off differently.

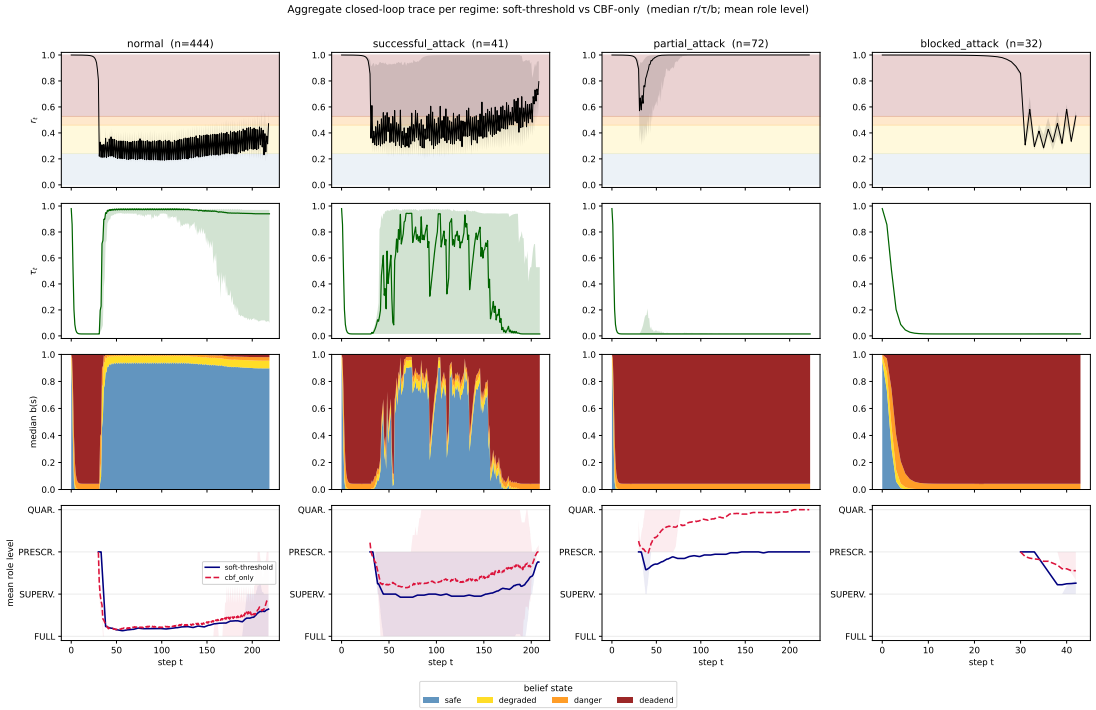

Figure 7.8.2: Aggregate offline-replay traces by regime (median r_t, τ_t, belief; mean role level), soft-threshold (solid) vs barrier-inspired filter (dashed). Benign episodes remain at full; attacks collapse trust and drive belief into deadend, with the safety filter restricting more aggressively.

Motivation behind MPC: cost regimes. HPR/BAR weight every block and allow equally and ignore latency while for a real deployment this is important. To show this, We define an operational cost that charges a miss (harmful action allowed, or false negative), a false restriction (benign action blocked, or false positive), and a latency term (steps until the harmful action is first blocked), then pick each controller's best operating point under a grid of cost regimes. The result (Figure 7.8.3) is a genuine crossover at a false-restriction cost ratio of ≈ 15: when false restrictions are relatively costly (utility-sensitive deployments), the graded soft/trust controller is cheaper; when a missed harmful action is catastrophic (security-critical deployments), the decisive safety filter is cheaper, and raising the latency weight shifts the advantage further toward the safety filter. The motivation for the MPC layer is therefore not a higher HPR, rather it is graded, auditable autonomy and a favourable cost profile in the utility-sensitive regime, rather than the safety filter's all or deny reaction.

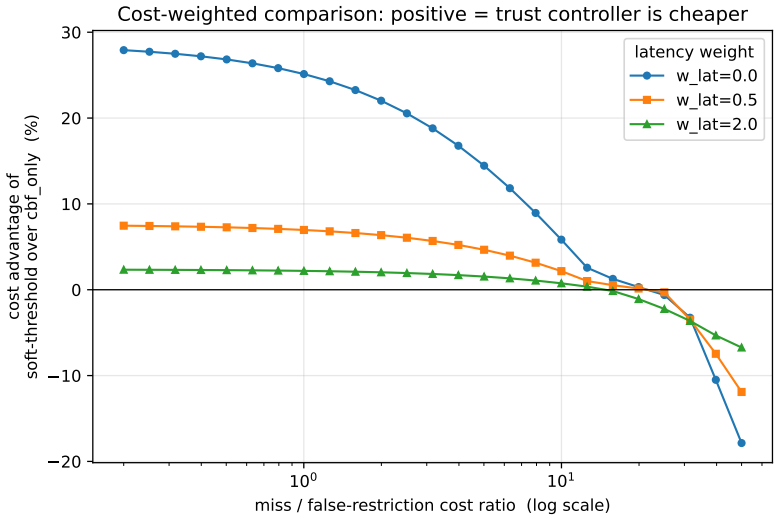

Figure 7.8.3: Cost-weighted comparison. Positive values mean the soft/trust controller is cheaper. There is a crossover near a miss/false-restriction ratio of 15; which controller is preferable depends on the deployment's cost regime.

Early detection: an honest null result. We tested whether the trust-integrating controller blocks harmful actions earlier on the successful-attack regime, where RQ1 showed instantaneous risk is least informative. Using a consistent definition (containment = the first step the harmful action is denied (when harm is actually stopped) the two controllers are identical: median time-to-first-block of 9 steps, block rate 0.317, and prevention-AUC 0.429 for both (Figure 7.8.4). The safety filter's only edge is reaching full quarantine faster (on 48.8% of episodes), but that quarantine is redundant for harm prevention because the block has already stopped the harm. We therefore do not claim that our controller detects attacks earlier in this experiment. As discussed previously, its main advantage is that it supports dynamic, tunable decisions based on the sensitivity and granularity of actions and roles.

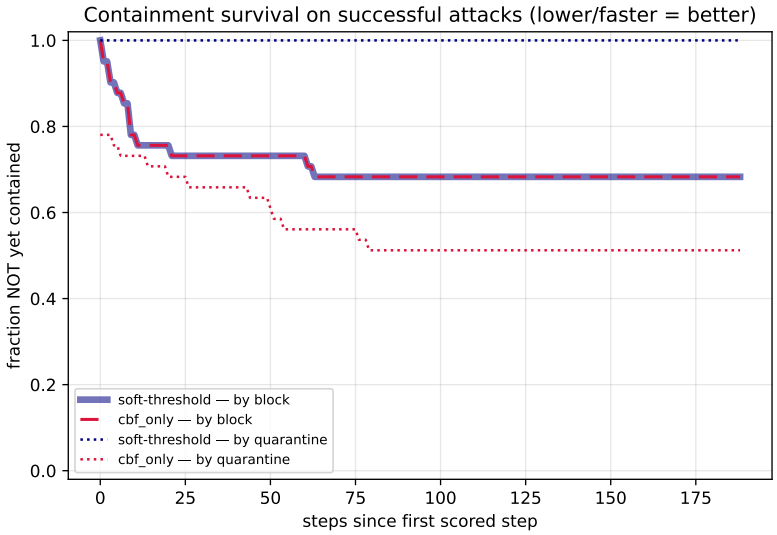

Figure 7.8.4: Containment survival on successful attacks. "By block" (harm stopped) is identical for both controllers (the curves coincide); only "by quarantine" (full isolation) separates them, and that does not reduce harm.

Several other limitations bound the strength of these conclusions.

Offline replay, not a true closed loop. The decisive limitation is that all controllers act on pre-recorded trajectories. The signals r_t, τ_t, b_t were generated policy collected the data, so a restriction the controller imposes does not actually change the agent's subsequent behaviour or its future risk. A real closed loop would feed each control action back into the environment, where blocking an action could divert, delay, or provoke the agent (effects this setup cannot observe). Consequently,, the latency and containment numbers should be read as relative comparisons under a fixed trace, not as guarantees of in place behaviour.

Single-agent decisions. Each decision concerns one agent in isolation, which is exactly the regime in which the quadratic MPC reduces to the linear rule. The cross-agent coupling that would motivate the quadratic form (Section 7.7) is therefore argued but not demonstrated.

Configuration and threshold sensitivity. Results depend on the chosen β, θ, and safety filter thresholds. Notably the soft policy never quarantines because θ_3 is unreachable by score; a different θ_3 or a trust-decay quarantine rule could change the MPC-vs-safety filter picture, and was not explored here.

Single environment and metric definitions. All episodes come from one simulated Secure Resource Grid, and the cost-regime conclusion depends on the particular cost model. The external validity of the crossover point in particular should be treated cautiously.

7.9 Integrated Pipeline Evaluation (RQ4)

The experimental pipeline integrates the three principal components of the proposed framework: the CVAE–LSTM produces an instantaneous behavioural-risk signal, the

belief-based trust filter accumulates this evidence over time, and the access-control policies map risk and trust to adaptive autonomy roles.

The integration is evaluated through offline replay rather than interactive closed-loop execution. Consequently, controller decisions do not influence subsequent observations or agent behaviour. RQ4 is therefore answered in terms of the compatibility of the implemented components and the resulting safety–utility trade-off under fixed trajectories, rather than in terms of closed-loop stability or deployment guarantees.

The results show that the integrated pipeline can retrospectively restrict a large proportion of sensitive actions occurring in attack episodes while preserving most sensitive actions in benign episodes. However, a true closed-loop evaluation remains necessary to determine how agents react to restrictions and how controller actions alter future risk and trust.

8 Conclusion and Future Work

This thesis began from a single observation: classical Identity and Access Management (IAM) was built for deterministic subjects whose behaviour is fully described by static credentials and whose authorization can be settled once, at a single point in time. Autonomous AI agents violate both assumptions. An agent authenticated with valid credentials can, within the same session, drift away from the intent under which it was authorized (e.g through prompt injection, memory poisoning, emergent reasoning errors, or cascading failures across collaborating agents) without any identifiable credential compromise. We called this the attribution gap: the actions an agent performs remain technically authorized, yet no longer reflect the intent that justified the authorization.

The central contribution of this work is a reformulation of IAM for autonomous agents from a discrete authorization decision into a closed-loop control process. Rather than asking once whether an identity may act, the proposed framework continuously monitors how an agent behaves, estimates risk and trust from that behaviour, and adapts the agent's privileges over time. Concretely, the framework couples three layers: a behavioural model that converts raw trajectories into a calibrated instantaneous risk signal (Section 7.3), a probabilistic trust filter that integrates that risk over time into a history-aware trust scalar (Section 7.4), and a control layer that turns risk and trust into a runtime access decision over an ordered set of autonomy levels (Section 7.5). This chapter consolidates what the empirical evaluation established about each layer, states the limitations that bound those conclusions, and sets out the directions that follow most directly from the results.

8.1 Summary and Conclusions

The main research question asked how IAM for autonomous agents can be reformulated as a closed-loop control problem in which privileges are continuously adapted to observed behaviour rather than fixed at authentication time. The threat model of Chapter 4, the framework of Chapter 5, the simulation environment and data-generation pipeline of Chapter 6, and the evaluation of Chapter 7 together answer that question in the affirmative, while also explain precisely where the reformulation pays off and where it does not. The four sub-questions are revisited below.

8.1.1 Behavioural representation as a calibrated risk signal (RQ1)

The first sub-question asked whether the runtime behaviour of an agent can be modelled as a stochastic, temporally evolving process suitable for online risk estimation, and to what extent a CVAE+LSTM architecture captures deviations from learned normal behaviour. The behavioural model, trained only on benign trajectories, learns a history-conditioned notion of normality and scores new behaviour by how poorly the model reconstructs it. On the primary evaluation regime (the union of successful and partial attacks, which are precisely the episodes a static

auditor failed to fully stop) the model attains a ROC-AUC of 0.888 and a PR-AUC of 0.937, a gain of $+0.243$ over a metadata-only baseline (0.645). This establishes that the behavioural trajectory does carry enough signal to separate manipulation from benign operation in real time, and that the signal is genuinely temporal: it emerges under windowed aggregation rather than at the single-step level, and a feature-group ablation confirmed that the detector does not rely on the visit-count intensities that are an artefact of how the attack library was written.

The more important conclusion, however, is a negative one that shaped the rest of the thesis. The instantaneous risk signal is not uniformly informative across attack classes. Partial attacks are detected cleanly (AUC 0.938) and even blocked attacks are detected well (AUC 0.853), but successful attacks (those that reach the adversarial objective with low auditor friction) are the hardest to detect (AUC 0.801) and overlap the benign distribution at the episode-mean level. The reconstruction-error score is therefore most discriminative on the attack class where dynamic IAM adds the least value (the auditor already stopped it) and least discriminative on the class where it must add the most. This inversion is the empirical core of the thesis: it is the quantitative evidence that point-in-time risk estimation is insufficient for regulating temporally evolving behaviour, and it is what motivates layering a temporal trust filter on top of the per-step signal.

8.1.2 Historical trust as a filtered belief (RQ2)

The second sub-question asked whether historical trust can be formalized as a belief over latent safety states using a POMDP, and whether such a formulation detects slow, subtle degradation that point-in-time scores miss. Modelling trust as a Bayes-filtered posterior over four hidden states ($\text{safe}, \text{degraded}, \text{danger}, \text{deadend}$) and reading trust as a weighted functional of that belief produced a signal that behaves as specified: it remains stable on benign trajectories (mean trust ≈ 0.81, standard deviation 0.03), collapses under sustained critical observations, and recovers from isolated noise spikes. Four of five trust-derived features outperform the instantaneous-risk baseline on the full task, and the mean-trust feature improves aggregate ROC-AUC from 0.876 to 0.890.

The decisive result for RQ2 is on the regime that RQ1 identified as the bottleneck. On successful attacks, the filtered final-step trust raises ROC-AUC from 0.751 to 0.805, a gain of $+0.054$, while leaving the easy regimes essentially unchanged. Temporal integration thus recovers part of the signal that the episode-mean risk dilutes, confirming that trust is most usefully treated as a filtered, history-aware quantity rather than a point statistic or a simple moving average. At the same time, the gain is partial rather than complete, and a principled fusion of the risk and trust signals did not improve on the single trust feature within the scope of this work; the controller therefore consumes both signals through a transparent heuristic. Closing the remaining gap on successful attacks is the most concrete open problem the thesis leaves behind, and it directly motivates the representation-learning direction.

8.1.3 Adaptive Access-Control Decisions under Offline Replay (RQ3)

The third sub-question asked how instantaneous behavioural risk and history-aware trust can be translated into adaptive autonomy roles, and how different role-selection

policies trade off harm-prevention rate against preservation of benign utility under offline replay. Three controllers were compared on identical replayed signals: a linear soft-threshold policy, a discrete finite-horizon quadratic-cost controller, and a barrier-inspired threshold safety filter.

The clearest finding is an equivalence rather than a victory. On a single agent with a small, discrete set of autonomy roles, the quadratic MPC provably and empirically reduces to the linear threshold rule: rounding a monotone scalar to the nearest of four roles is identical to placing three thresholds on a suspicion signal, and the two controllers differ by -0.0038 in harm-prevention rate across the matched frontier while the discrete finite-horizon optimization costs roughly three orders of magnitude more compute per decision. The barrier-inspired threshold safety filter, in turn, captures almost all of the achievable safety/utility trade-off by itself, leading the soft policy by only ≈ 0.01 in harm-prevention rate. A deliberately honest null result reinforces the point: the trust-integrating controller did not block harmful actions earlier than the memoryless safety net on successful attacks (both reach a median time-to-first-block of nine steps with identical prevention-AUC).

The conclusion is therefore nuanced. The predictive, optimization-based formulation does not buy higher detection or earlier containment in the single-agent setting studied here; its value lies elsewhere—in graded, auditable, operator-tunable autonomy, and in a favourable cost profile in utility-sensitive deployments, where a cost-weighted comparison shows the graded controller is cheaper than the threshold safety filter until false restrictions become roughly fifteen times less costly than missed harm. Crucially, the analysis also identifies exactly when the quadratic form becomes necessary rather than equivalent: as soon as decisions couple across multiple interacting agents, the objective acquires a non-diagonal structure that no per-agent threshold rule can express. This is argued in the thesis but, by construction of the single-agent experiments, not yet demonstrated—and it is the cleanest entry point to the multi-agent work in Section 8.3.1.

8.1.4 Integrated Experimental Evaluation (RQ4)

The pipeline retrospectively denies a large proportion of sensitive actions occurring in attack-labelled episodes (HPR ≈ 0.94) while preserving most sensitive actions in benign episodes (BAR ≈ 0.75–0.80).

However, controller decisions do not alter subsequent observations or agent behaviour. The experiment therefore demonstrates component compatibility and comparative fixed-trajectory performance, but it does not establish a true feedback loop, closed-loop stability, or deployment effectiveness. Interactive closed-loop validation remains necessary.

Taken together, the four answers substantiate the central claim of the thesis: access control for autonomous agents can be reformulated as a closed-loop control problem, and doing so is both feasible and informative. It is feasible in that each component admits a concrete, calibrated, online implementation; it is informative in that the reformulation makes explicit where behavioural monitoring helps—continuous, trajectory-level signals recover attacks that a single static check misses—and where it does not—a stealthy attack that never deviates step by step remains the hard case, and

predictive control adds auditability and graded response rather than raw detection power in the single-agent regime. Both the positive and the negative findings are contributions: the framework demonstrates a working bridge between deep probabilistic behavioural modelling and control theory for IAM, and the evaluation delivers a reproducible environment, a labelled dataset of benign and adversarial agent trajectories, and a set of baselines and metrics against which future behaviour-aware IAM can be measured.

8.2 Limitations

The conclusions above are bounded by the design choices that made a controlled, reproducible study possible. They are restated here so that the strength of each claim is read against its proper scope.

Evaluation under offline replay. The most consequential limitation is that the controllers act on pre-recorded trajectories. Because the risk, trust, and belief signals were produced by the data-collection policy, a restriction the controller imposes does not actually alter the agent's subsequent behaviour or its future risk. A genuine closed loop would feed each control action back into the environment, where blocking an action could divert, delay, or provoke the agent.

Single-agent decisions. Every decision concerns one agent in isolation, which is exactly the regime in which the discrete finite-horizon controller collapses to the linear rule. The cross-agent coupling that would justify the full optimization (shared resource budgets, correlated escalations, limits on how many agents may simultaneously hold high autonomy) is argued but not exercised.

A single, simplified environment. All episodes originate from one lightweight, graph-structured Secure Resource Grid of nine nodes across three sensitivity zones, chosen for tractability and reproducibility. The simulator-to-field gap is real in a way that the abstraction omits the richness of production tool ecosystems and enterprise environments.

8.3 Future Work

The directions for the future contributions that follow are ordered roughly by how directly they address the limitations just stated.

8.3.1 Multi-agent control and cross-agent coupling

The control analysis showed that, for one agent over a small discrete role set, the predictive optimization is indistinguishable from a linear threshold and a plain safety net. It also identified exactly when that equivalence breaks (in a genuine multi-agent deployment, decisions couple. Limiting the number of agents that may simultaneously hold high autonomy, penalizing correlated escalations that together exceed a shared resource budget, or containing a cascading failure that propagates between collaborating agents all introduce cross-agent terms that no per-agent rule can represent) . Extending the framework to a population of interacting agents (with a joint state, coupled constraints, and an objective) would let the predictive controller demonstrate the advantage that is currently only argued, and would directly exercise

the cascading-failure attack class that the single-agent setup could not fully test. This is also the natural setting in which to study agent-to-agent trust propagation (how the trust collapse of one agent should, or should not, lower the privileges of those it communicates with).

8.3.2 Closing the loop and moving toward deployment

The most important methodological step is to convert the offline replay into a true closed loop, in which each control action is applied inside the environment and the agent's subsequent behaviour, risk, and trust reflect the restriction. Only then can the framework's containment and latency be measured as behaviour rather than as a relative ranking over a fixed trace, and only then can effects that the present setup cannot observe be studied. Beyond closing the loop in simulation, transferring the framework to a realistic deployment, with production tools and real workloads, would test the simulator-to-field gap that the single environment leaves open.

8.3.3 Richer simulation and cyber-physical environments

The nine-node grid was deliberately minimal. Scaling the environment toward larger and more heterogeneous resource graphs, real tool integrations, and longer horizons would stress the behavioural model on the kind of diversity and scale a real system exhibits. A richer environment is also the prerequisite for a more demanding adversarial library: more attack types, more episodes per type, and in particular attacks engineered to stay on the benign manifold.

8.3.4 Robustness to adaptive adversaries

The threat model assumes attacks drawn from a fixed library rather than an adversary that adapts to the defence. Because the predictive controller relies on a forward model and the detector on a learned notion of normality, both present a surface an adaptive attacker could get advantage from by shaping behaviour to keep risk below threshold, mimicking benign trajectories to suppress reconstruction error, or exploiting the dynamics model the controller plans against. Studying the framework under adaptive, defence-aware adversaries is necessary before any security guarantee can be claimed.

8.3.5 Toward a complete agentic IAM stack

Finally, the behavioural and control-theoretic contribution of this thesis was deliberately separated from the cryptographic and infrastructural layers of a full agentic IAM architecture. Integrating the runtime governance layer developed here with the identity, credential, and token mechanisms it presupposes would situate behaviour-aware control within an end-to-end system, in which the dynamic privilege decisions produced by the controller are enforced by the same machinery that issues and verifies agent credentials.

8.4 Concluding Remarks

Autonomous agents break the assumption that authorization can be settled once, because what an agent is permitted to do and what it actually does can diverge long

after authentication. This thesis treated that divergence as a control problem rather than a one-time decision, and showed that an agent's behavioural trajectory carries enough signal to estimate risk and trust online and to adapt its privileges accordingly. The framework is not a finished defence for the reasons mentioned above. But the reformulation itself is the lasting contribution. By casting identity and access management for autonomous agents as continuous, behaviour-driven control, it turns a static gatekeeping question into a dynamic one, and in doing so makes visible both what such a system can already do and exactly where the next work must focus.

Bibliography

[1] Sahaya Jestus Lazer, Kshitiz Aryal, Maanak Gupta, et al. *A Survey of Agentic AI and Cybersecurity: Challenges, Opportunities and Use-case Prototypes*. Manuscript submitted to ACM. 2026. arXiv: 2601.05293 [cs.CR].

[2] OWASP GenAI Security Project. *OWASP Top 10 for Agentic Applications 2026*. https://genai.owasp.org. Version 2026. Dec. 2025.

[3] Mohamad Abou Ali, Fadi Dornaika, and Jinan Charafeddine. "Agentic AI: a comprehensive survey of architectures, applications, and future directions". In: *Artificial Intelligence Review* 59 (2026), p. 11. doi: 10.1007/s10462-025-11422-4.

[4] Madan Baduwal and Priyanka Paudel. *Evaluating Agentic Artificial Intelligence: A Comprehensive Survey of Metrics, Benchmarks, and Methodologies*. Manuscript. 2025.

[5] Jinyuan Fang, Yanwen Peng, Xi Zhang, et al. "A Comprehensive Survey of Self-Evolving AI Agents: A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems". In: *arXiv preprint arXiv:2508.07407* (2025).

[6] Muthukrishnan Manoharan. "Securing Autonomous AI Agents in Hybrid Enterprise Systems: A Behavior-Centric Trust Enforcement Architecture". In: *European Modern Studies Journal* 9.4 (2025), pp. 308–325. doi: 10.59573/emsj.9(4).2025.32.

[7] Sahaya Jestus Lazer, Kshitiz Aryal, Maanak Gupta, et al. *A Survey of Agentic AI and Cybersecurity: Challenges, Opportunities and Use-case Prototypes*. 2026. arXiv: 2601.05293 [cs.CR].

[8] Angelika Steinacker and Hari Hayagreevan. *Governing AI Agents: An Agent-Aware IAM Framework*. Tech. rep. Licensed under CC BY 4.0. Independent Publication, 2026.

[9] Aditi S. Krishnapriyan et al. *A Comprehensive Review of AI Agents: Transforming Possibilities in Technology and Beyond*. Manuscript. 2025.

[10] Jingtao Ding, Yunke Zhang, Yu Shang, et al. "Understanding World or Predicting Future? A Comprehensive Survey of World Models". In: *ACM Computing Surveys* 58.3 (2025), 57:1–57:38. doi: 10.1145/3746449.

[11] Conor Bronsdon. *Real-Time Anomaly Detection for Multi-Agent AI Systems*. Accessed: 2026-02-21. Galileo. June 2025. url: https://galileo.ai/blog/real-time-anomaly-detection-multi-agent-ai.

[12] Cassiano Moralles, Luis Antonio L. F. da Costa, Sandro José Rigo, et al. "A Systematic Literature Review of Agentic AI: Definitions, Architectures, and Challenges". In: *IEEE Access* (2026). doi: 10.1109/ACCESS.2026.3668138.

[13] Bang Liu, Xinfeng Li, Jiayi Zhang, et al. "Advances and Challenges in Foundation Agents: From Brain-Inspired Intelligence to Evolutionary, Collaborative, and Safe Systems". In: *arXiv preprint arXiv:2504.01990* (2025).

[14] Rahul Karne. "Embodied AI for Security: Robotic Agents in Surveillance, Disaster Response, and Cyber-Physical Defense". In: *2025 10th IEEE*

International Conference on Smart Structures and Systems (ICSSS). 2025. doi: 10.1109/ICSSS66939.2025.11346155.

[15] David Temoshok, Diana Proud-Madruga, Yee-Yin Choong, et al. *Digital Identity Guidelines*. Tech. rep. NIST SP 800-63-4 2pd. National Institute of Standards and Technology, Aug. 2024. doi: 10.6028/NIST.SP.800-63-4.2pd. url: https://doi.org/10.6028/NIST.SP.800-63-4.2pd.

[16] Ken Huang and Cloud Security Alliance. *Agentic AI Identity Management Approach*. Accessed: 2026-02-16. Mar. 2025. url: https://cloudsecurityalliance.org/blog/2025/03/11/agentic-ai-identity-management-approach.

[17] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, et al. "Role-Based Access Control Models". In: *Computer* 29.2 (Feb. 1996), pp. 38–47.

[18] CrowdStrike. *What Are Non-Human Identities (NHIs)?* Accessed: 2026-02-16. Jan. 2025. url: https://www.crowdstrike.com/en-us/cybersecurity-101/identity-protection/non-human-identities/.

[19] IBM. *Introducing Machine Identity Management to strengthen IAM for non-human identities*. Accessed: 2026-02-16. Nov. 2025. url: https://www.ibm.com/new/announcements/introducing-machine-identity-management-to-strengthen-iam-for-non-human-identities.

[20] David Temoshok, Ryan Galluzzo, Connie LaSalle, et al. *Digital Identity Guidelines*. Tech. rep. NIST SP 800-63-4. National Institute of Standards and Technology, July 2025. doi: 10.6028/NIST.SP.800-63-4. url: https://doi.org/10.6028/NIST.SP.800-63-4.

[21] HYPR. *NIST SP 800-63-3 Digital Identity Guidelines — Review*. Accessed: 2026-02-16. 2024. url: https://www.hypr.com/blog/nist-sp-800-63-3-digital-identity-guidelines-review.

[22] Abhishek Goswami. *Agentic JWT: A Secure Delegation Protocol for Autonomous AI Agents*. 2025. arXiv: 2509.13597 [cs.CR].

[23] Hany F. Atlam, Muhammad Ajmal Azad, Madini O. Alassafi, et al. "Risk-Based Access Control Model: A Systematic Literature Review". In: *Future Internet* 12.6 (2020), p. 103. doi: 10.3390/fi12060103.

[24] Vincent C. Hu, D. Richard Kuhn, and David F. Ferraiolo. "Attribute-Based Access Control". In: *Computer* 48.2 (Feb. 2015), pp. 85–88.

[25] Shiyu Xiao, Yuhang Ye, Nadia Kanwal, et al. "SoK: Context and Risk Aware Access Control for Zero Trust Systems". In: *Security and Communication Networks* 2022 (2022), p. 7026779. doi: 10.1155/2022/7026779.

[26] Scott Rose, Oliver Borchert, Stu Mitchell, et al. *Zero Trust Architecture*. Tech. rep. NIST Special Publication 800-207. National Institute of Standards and Technology, Aug. 2020. doi: 10.6028/NIST.SP.800-207.

[27] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. Tech. rep. NIST AI 100-1. NIST, Jan. 2023. doi: 10.6028/NIST.AI.100-1. url: https://doi.org/10.6028/NIST.AI.100-1.

[28] Amazon Web Services. *The Agentic AI Security Scoping Matrix: A Framework for Securing Autonomous AI Systems*. Accessed: 2026-02-22. Nov. 2025. url: https://aws.amazon.com/blogs/security/the-agentic-ai-security-scoping-matrix-a-framework-for-securing-autonomous-ai-systems/.

[29] Josh Woodruff. *The Agentic Trust Framework: Zero Trust Governance for AI Agents*. Accessed: 2026-02-22. Cloud Security Alliance. Feb. 2026. url: https://cloudsecurityalliance.org/blog/2026/02/02/the-agentic-trust-framework-zero-trust-governance-for-ai-agents.

[30] Charles L. Wang, Trisha Singhal, Ameya Kelkar, et al. *MI9: An Integrated Runtime Governance Framework for Agentic AI*. 2025. arXiv: 2508.03858 [cs.AI].

[31] Tianneng Shi, Jingxuan He, Zhun Wang, et al. *Progent: Programmable Privilege Control for LLM Agents*. 2025. arXiv: 2504.11703 [cs.CR].

[32] E. Jeong and D. Y. "A Trust Score-Based Access Control Model for Zero Trust Architecture: Design, Sensitivity Analysis, and Real-World Performance Evaluation". In: *Applied Sciences* 15 (2025), p. 9551. doi: 10.3390/app15179551.

[33] Daniel Ricardo dos Santos, Roberto Marinho, Gustavo Roecker Schmitt, et al. "A framework and risk assessment approaches for risk-based access control in the cloud". In: *Journal of Network and Computer Applications* 74 (2016), pp. 86–97. doi: 10.1016/j.jnca.2016.08.013.

[34] Sudip Chakraborty and Indrajit Ray. "TrustBAC: Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems". In: *Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies*. ACM, 2006, pp. 49–58. isbn: 1-59593-353-0. doi: 10.1145/1133058.1133067.

[35] Hany F. Atlam, Ahmed Alenezi, Robert J. Walters, et al. "Developing an Adaptive Risk-Based Access Control Model for the Internet of Things". In: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2017, pp. 655–661. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.103.

[36] Xu Jing, Zhengnan Liu, Shuqin Li, et al. "A cloud-user behavior assessment based dynamic access control model". In: *International Journal of System Assurance Engineering and Management* 8.Suppl. 3 (2017), S1966–S1975. doi: 10.1007/s13198-015-0411-1.

[37] Simone Fischer-Hübner, Costas Lambrinoudakis, Gabriele Kotsis, et al., eds. *Trust, Privacy and Security in Digital Business: 18th International Conference, TrustBus 2021, Virtual Event, September 27–30, 2021, Proceedings*. Vol. 12927. Lecture Notes in Computer Science. Springer, 2021. isbn: 978-3-030-86586-3. doi: 10.1007/978-3-030-86586-3.

[38] Zeinab M. Iqal and Ali Selamat. "A Comprehensive Analysis of Risk-Based Access Control Models for IoT: Balancing Security, Adaptability, and Resource Efficiency". In: *2024 IEEE International Conference on Computing (ICOCO)*. IEEE, 2024, pp. 344–348. doi: 10.1109/ICOCO62848.2024.10928193.

[39] Xiaoya Cao, Zhenya Chen, Ming Yang, et al. "Zero Trust-Based Dynamic and Continuous Access Control for Mobile Devices". In: *2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2025. doi: 10.1109/Trustcom66490.2025.00344.

[40] Annanda Thavymony Rath and Jean-Noel Colin. “Adaptive Risk-Aware Access Control Model for Internet of Things”. In: *2017 International Workshop on Secure Internet of Things*. IEEE, 2017. doi: 10.1109/SIoT.2017.00010.

[41] Hany F. Atlam, Ahmed Alenezi, Raid Khalid Hussein, et al. “Validation of an Adaptive Risk-based Access Control Model for the Internet of Things”. In: *International Journal of Computer Network and Information Security* 10.1 (2018), pp. 26–35. doi: 10.5815/ijcnis.2018.01.04.

[42] Himanshu Sharma. “Behavioral Analytics and Zero Trust”. In: *International Journal of Information Technology and Management Information Systems* 12.1 (2021), pp. 63–84.

[43] Abhishek Tripathi, Kumar Rajan, Vishwajit Kumar, et al. “Real Time Adaptive Access Control with Behavioral Analytics for Enhanced Cybersecurity in IoT and Cloud Systems”. In: *Proceedings of the Ninth International Conference on Research in Intelligent Computing in Engineering*. 2024, pp. 151–155. doi: 10.15439/2024R75.

[44] Wawan Yunanto and Hsing-Kuo Pao. “User Behaviour Risk Evaluation in Zero Trust Architecture Environment”. In: *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*. IEEE, 2022. doi: 10.1109/WF-IOT54382.2022.10152197.

[45] A. Riyaz Fathima and A. Saravanan. “An approach to cloud user access control using behavioral biometric-based authentication and continuous monitoring”. In: *International Journal of Advanced Technology and Engineering Exploration* 11.119 (2024), pp. 1469–1497. doi: 10.19101/IJATEE.2024.111100516.

[46] Stellar Cyber. *Top Agentic AI Security Threats in 2026*. Accessed: 2026-02-21. 2026. url: https://stellarcyber.ai/learn/agentic-ai-securiry-threats/.

[47] Pallavi Zambare, Venkata Nikhil Thanikella, and Ying Liu. *Securing Agentic AI: Threat Modeling and Risk Analysis for Network Monitoring Agentic AI System*. 2025.

[48] Gravitee. *The State of AI Agent Security 2026*. Tech. rep. Survey report. Gravitee, 2026.

[49] InstaTunnel. *Agentic Memory Poisoning: How Long-Term AI Context Can Be Weaponized*. Accessed: 2026-02-21. Jan. 2026. url: https://medium.com/@instatunnel/agentic-memory-poisoning-how-long-term-ai-context-can-be-weaponized-7c0eb213bd1a.

[50] Unit 42. *Indirect Prompt Injection Poisons AI Long-Term Memory*. Accessed: 2026-02-22. 2025. url: https://unit42.paloaltonetworks.com/indirect-prompt-injection-poisons-ai-longterm-memory/.

[51] Khmaïess Al Jannadi. *Prompt Injection Attacks in Large Language Models: Vulnerabilities, Exploitation Techniques, and Defense Strategies*. Accessed: 2026-02-21. Jan. 2026. url: https://medium.com/@jannadikhemais/prompt-injection-attacks-in-large-language-models-vulnerabilities-exploitation-techniques-and-e00fe683f6d7.

[52] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. “A Descriptive Study of Microsoft’s Threat Modeling Technique”. In: *Requirements Engineering* 20.2 (2015), pp. 163–180. doi: 10.1007/s00766-013-0195-2.

[53] Adam Shostack. “Experiences Threat Modeling at Microsoft”. In: *Workshop on Modeling Security (ModSec)*. 2008.

[54] Nataliya Shevchenko, Timothy A. Chick, Paige O'Riordan, et al. *Threat Modeling: A Summary of Available Methods*. Tech. rep. Software Engineering Institute, Carnegie Mellon University, July 2018.

[55] Ken Huang and Chris Hughes. *Securing AI Agents: Foundations, Frameworks, and Real-World Deployment*. Springer Cham, 2025. isbn: 978-3-032-02130-4. doi: 10.1007/978-3-032-02130-4.

[56] Majid Mollaeefar, Andrea Bissoli, Dimitri Van Landuyt, et al. *PILLAR: LINDDUN Privacy Threat Modeling using LLMs*. Manuscript. 2024.

[57] Qiusi Zhan, Zhixiang Liang, Zifan Ying, et al. "INJECAGENT: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents". In: *Findings of the Association for Computational Linguistics: ACL 2024*. 2024, pp. 10471–10506.

[58] Paul Theurich, Josepha Witt, and Sebastian Richter. "Practices and challenges of threat modelling in agile environments". In: *Informatik Spektrum* 46 (2023), pp. 220–229. doi: 10.1007/s00287-023-01549-5.

[59] Christopher Alberts, Audrey Dorofee, John Stevens, et al. *OCTAVE Method Implementation Guide, Version 2.0*. Tech. rep. Software Engineering Institute, Carnegie Mellon University, 1999.

[60] Paul Saitta, Brenda Larcom, and Michael Eddington. *Trike v.1 Methodology Document [Draft]*. OctoTrike. Draft. July 2005.

[61] Norm Hardy. *The Confused Deputy*. Technical note / article. Also circulated via KeyKOS / Key Logic. 1988.

[62] Martín Abadi, Michael Burrows, Butler Lampson, et al. "A Calculus for Access Control in Distributed Systems". In: *ACM Transactions on Programming Languages and Systems* 15.3 (Sept. 1993), pp. 706–734.

[63] Abdullah Abdulaziz, Syed Rafiul Hussain, and Elisa Bertino. "Asset-Centric Threat Modeling for Connected Medical Devices". In: *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2024.

[64] Daniel Fett, Ralf Küsters, and Guido Schmitz. "A Comprehensive Formal Security Analysis of OAuth 2.0". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, 2016, pp. 1204–1215. isbn: 978-1-4503-4139-4. doi: 10.1145/2976749.2978385.

[65] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. "Empirical evaluation of a privacy-focused threat modeling methodology". In: *The Journal of Systems and Software* 96 (2014), pp. 122–138. doi: 10.1016/j.jss.2014.05.075.

[66] MITRE. *advmlthreatmatrix: Adversarial Threat Landscape for AI Systems*. GitHub repository. Now released under the ATLAS branding. 2020. url: https://github.com/mitre/advmlthreatmatrix.

[67] Apostol Vassilev, Alina Oprea, Alie Fordyce, et al. *Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations*. Tech. rep. NIST AI 100-2e2023. National Institute of Standards and Technology, Jan. 2024. doi: 10.6028/NIST.AI.100-2e2023.

[68] Elham Tabassi. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. Tech. rep. NIST AI 100-1. National Institute of Standards and Technology, Jan. 2023. doi: 10.6028/NIST.AI.100-1.

[69] Vineeth Sai Narajala and Om Narayan. *Securing Agentic AI: A Comprehensive Threat Model and Mitigation Framework for Generative AI Agents*. 2025.

[70] Dave Clever. *Adversarial Threat Modeling for Large-Scale AI Systems in Enterprise Networks*. Manuscript. 2025.

[71] Herman Errico, Jiquan Ngiam, and Shanita Sojan. "Securing the Model Context Protocol (MCP): Risks, Controls, and Governance". In: *arXiv preprint arXiv:2511.20920* (2025). doi: 10.48550/arXiv.2511.20920.

[72] OWASP GenAI Security Project - Agentic Security Initiative. *Multi-Agentic System Threat Modelling Guide*. OWASP. Version 1.0. Apr. 2025.

[73] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and Acting in Partially Observable Stochastic Domains". In: *Artificial Intelligence* 101.1–2 (1998), pp. 99–134. doi: 10.1016/S0004-3702(98)00023-X.

[74] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations (ICLR)*. arXiv:1312.6114. 2014.

[75] Jinwon An and Sungzoon Cho. *Variational Autoencoder based Anomaly Detection using Reconstruction Probability*. Special Lecture on IE 1. SNU Data Mining Center, 2015, pp. 1–18.

[76] Zhuxi Jiang, Yin Zheng, Huachun Tan, et al. "Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 1965–1972.

[77] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[78] Junyoung Chung, Kyle Kastner, Laurent Dinh, et al. "A Recurrent Latent Variable Model for Sequential Data". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 28. arXiv:1506.02216. 2015, pp. 2980–2988.

[79] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 28. 2015, pp. 3483–3491.

[80] Shuyu Lin, Ronald Clark, Robert Birke, et al. "Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 4322–4326. doi: 10.1109/ICASSP40776.2020.9053558.

[81] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd ed. Nob Hill Publishing, 2017. isbn: 978-0-9759377-3-0.

[82] Farama Foundation. *Gymnasium Documentation*. https://gymnasium.farama.org/. 2023.

[83] LangChain Team. *LangGraph Documentation*. https://langchain-ai.github.io/langgraph/. 2024.

[84] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Vol. 70. Proceedings of Machine Learning Research (PMLR). 2017, pp. 3319–3328.

A Experimental Implementation and Mathematical Specification

Chapter 5 presents the general control-theoretic framework proposed in this thesis. The present appendix documents the narrower implementation used in the experiments. Its purpose is reproducibility: it specifies the tensor representation, neural-network architecture, loss function, risk calibration, belief-state trust filter, role-assignment policies, and offline-replay metrics exactly as implemented in the analysis notebooks.

The implementation is intentionally simpler than the general framework. In particular, the trust component performs action-independent Bayesian filtering rather than solving a complete POMDP policy, the controller selects among four predefined autonomy roles rather than optimising an arbitrary permission vector, and controller decisions are evaluated retrospectively on fixed trajectories rather than being fed back into the simulation environment.

A.1 Relation to the Proposed Framework

The principal simplifications are summarised in Table A.1.1.

Table A.1.1: Mapping from the general framework of Chapter 5 to the experimental implementation.

Component	General formulation	Experimental implementation
Behavioural model	Conditional recurrent latent-variable model	Fixed CVAE–LSTM with 21 inputs, a 64-dimensional recurrent state, and a 16-dimensional latent variable
Trust estimation	POMDP with potentially action-dependent transition and observation kernels	Action-independent Bayesian belief filter over four safety states
Access decision	Predictive optimisation of autonomy and permissions	Selection among four predefined autonomy roles using risk and trust
Safety layer	Barrier-function constraints	Barrier-inspired threshold safety filter
Evaluation	Closed-loop interaction with the environment	Offline replay of trajectories collected under full privilege

A.2 Input Representation and Episode Splits

At each valid step, the model receives a 21-dimensional vector

$$x_t = \begin{bmatrix} (x_t^{\text{cont}})^\top & a_t & d_t \end{bmatrix}^\top \in \mathbb{R}^{21}, \tag{A.2.1}$$

where $x_t^{\text{cont}} \in \mathbb{R}^{19}$ contains the continuous observation, counter, timing, and telemetry channels, a_t is the action identifier, and d_t is the auditor-decision identifier. The categorical identifiers are represented as integer classes. After shifting the padding sentinel to a non-negative index, the action and auditor heads contain six and three classes, respectively.

Only the 19 continuous channels are standardised. Let μ_j and σ_j be the preprocessing statistics for continuous feature j. The standardised input is

$$\tilde{x}_{t,j}^{\text{cont}} = \frac{x_{t,j}^{\text{cont}} - \mu_j}{\sigma_j}, \qquad j = 1, \dots, 19. \tag{A.2.2}$$

The categorical identifiers are passed through without standardisation. Padded episode positions are recorded by a mask $m_t \in \{0, 1\}$.

The 444 valid normal episodes are divided at episode level using a fixed random seed into 310 training episodes, 66 validation episodes, and 68 normal test episodes.

Training windows have length

$$W = 32 \tag{A.2.3}$$

and are extracted with stride 8. Episode-level reconstruction scores are computed with stride 16, whereas the temporal trust analysis uses stride-one windows to produce one risk value per recorded step.

A.3 Implemented CVAE–LSTM Architecture

For a window

$$X_t = \begin{bmatrix} x_{t-W+1} & \cdots & x_t \end{bmatrix}^\top \in \mathbb{R}^{W\times 21}, \tag{A.3.1}$$

the LSTM produces a sequence of recurrent states

$$h_1, \dots, h_W = \mathbf{LSTM}(X_t), \qquad h_j \in \mathbb{R}^{64}. \tag{A.3.2}$$

The conditioning state is shifted by one step so that the latent representation at position j uses only the preceding recurrent context:

$$h_{j-1}^{\text{cond}} = \begin{cases} 0, & j = 1, \\ h_{j-1}, & j > 1. \end{cases} \tag{A.3.3}$$

The encoder concatenates the current input and previous recurrent state:

$$e_j = \mathrm{Dropout}_{0.2}\left(\mathrm{ELU}\left(W_e \begin{bmatrix} x_j \\ h_{j-1}^{\text{cond}} \end{bmatrix} + b_e\right)\right), \qquad e_j \in \mathbb{R}^{64}, \tag{A.3.4}$$

$$\mu_j = W_\mu e_j + b_\mu, \tag{A.3.5}$$

$$\log \sigma_j^2 = \mathrm{clip}(W_\sigma e_j + b_\sigma, -6, 2), \tag{A.3.6}$$

where $\mu_j, \log\sigma_j^2 \in \mathbb{R}^{16}$. A latent sample is obtained by the reparameterisation rule

$$z_j = \mu_j + \exp\left(\frac{1}{2}\log\sigma_j^2\right) \odot \epsilon_j, \qquad \epsilon_j \sim \mathcal{N}(0, I_{16}). \tag{A.3.7}$$

The decoder receives the concatenated latent and recurrent vectors:

$$g_j^{(1)} = \mathrm{Dropout}_{0.2}\left(\mathrm{ELU}\left(W_{d,1} \begin{bmatrix} z_j \\ h_{j-1}^{\text{cond}} \end{bmatrix} + b_{d,1}\right)\right), \tag{A.3.8}$$

$$g_j^{(2)} = \mathrm{ELU}\left(W_{d,2} g_j^{(1)} + b_{d,2}\right), \qquad g_j^{(2)} \in \mathbb{R}^{64}. \tag{A.3.9}$$

Three output heads reconstruct the mixed feature types:

$$\hat{x}_j^{\text{cont}} = W_c g_j^{(2)} + b_c \in \mathbb{R}^{19}, \tag{A.3.10}$$

$$\ell_j^a = W_a g_j^{(2)} + b_a \in \mathbb{R}^6, \tag{A.3.11}$$

$$\ell_j^d = W_d g_j^{(2)} + b_d \in \mathbb{R}^3. \tag{A.3.12}$$

Here ℓ_j^a and ℓ_j^d are categorical logits. The complete tensor flow is shown in Figure A.3.1.

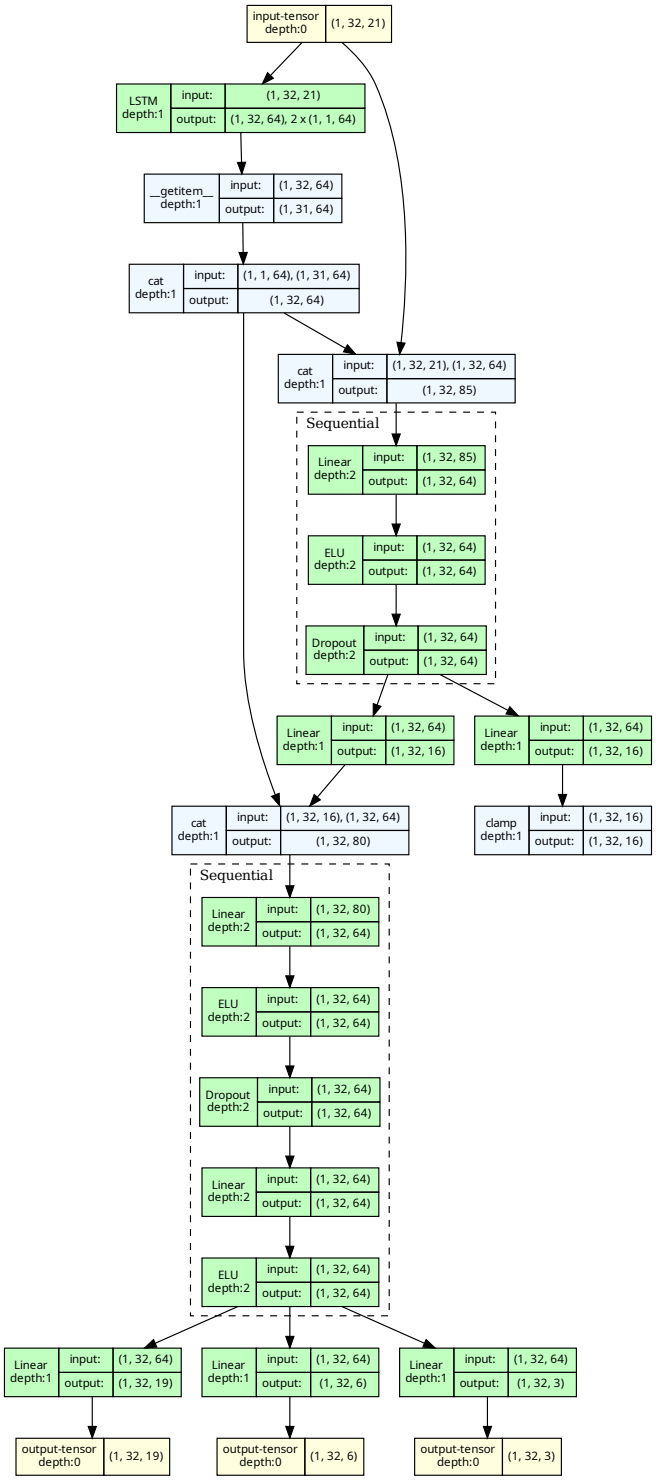

Figure A.3.1: Implemented CVAE–LSTM architecture for a batch containing one 32-step window. The input tensor has shape $(1, 32, 21)$, the recurrent state has dimension 64, the latent variable has dimension 16, and the decoder produces continuous, action, and auditor outputs of dimensions 19, 6, and 3, respectively.

A.4 Latent Priors and Training Objective

Two latent priors are evaluated. The baseline prior is

$$p_{\mathrm{N}}(z) = \mathcal{N}(0, I_{16}). \tag{A.4.1}$$

The multimodal alternative is an eight-component Gaussian mixture,

$$p_{\mathrm{GMM}}(z) = \sum_{k=1}^{8} \pi_k \, \mathcal{N}\big(z; \mu_k, \mathrm{diag}(\sigma_k^2)\big), \qquad \sum_{k=1}^{8} \pi_k = 1. \tag{A.4.2}$$

For the standard-normal prior, the Kullback–Leibler term has the usual closed form. For the mixture prior it is approximated using the reparameterised sample z_j:

$$\widehat{D}_{\mathrm{KL},j} = \log q_\phi(z_j \mid x_j, h_{j-1}^{\mathrm{cond}}) - \log p_{\mathrm{GMM}}(z_j). \tag{A.4.3}$$

For each valid position, the reconstruction term is

$$\rho_j = \big\| \hat{x}_j^{\mathrm{cont}} - x_j^{\mathrm{cont}} \big\|_2^2 + \mathrm{CE}(\ell_j^a, a_j) + \mathrm{CE}(\ell_j^d, d_j). \tag{A.4.4}$$

For a mini-batch with validity mask $m_{b,j}$, the four loss components are

$$\mathcal{L}_{\mathrm{cont}} = \frac{\sum_{b,j} m_{b,j} \big\| \hat{x}_{b,j}^{\mathrm{cont}} - x_{b,j}^{\mathrm{cont}} \big\|_2^2}{\sum_{b,j} m_{b,j}}, \tag{A.4.5}$$

$$\mathcal{L}_a = \frac{\sum_{b,j} m_{b,j} \, \mathrm{CE}(\ell_{b,j}^a, a_{b,j})}{\sum_{b,j} m_{b,j}}, \tag{A.4.6}$$

$$\mathcal{L}_d = \frac{\sum_{b,j} m_{b,j} \, \mathrm{CE}(\ell_{b,j}^d, d_{b,j})}{\sum_{b,j} m_{b,j}}, \tag{A.4.7}$$

$$\mathcal{L}_{\mathrm{KL}} = \frac{\sum_{b,j} m_{b,j} D_{\mathrm{KL},b,j}}{\sum_{b,j} m_{b,j}}. \tag{A.4.8}$$

The optimised objective at epoch e is

$$\mathcal{J}_e = \mathcal{L}_{\mathrm{cont}} + \mathcal{L}_a + \mathcal{L}_d + \beta_e \mathcal{L}_{\mathrm{KL}}, \tag{A.4.9}$$

with the warm-up schedule

$$\beta_e = \min\left(1, \frac{e+1}{10}\right). \tag{A.4.10}$$

The model is trained for 300 epochs using Adam with learning rate 10^{-3}, batch size 128, and gradient-norm clipping at 5. The retained checkpoint is the one with the smallest validation objective. The GMM-prior model is used in the downstream analysis following an exploratory comparison of primary-regime ROC–AUC.

A.5 Reconstruction Scores and Risk Calibration

A.5.1 Episode-level reconstruction score

During episode-level evaluation, windows of length 32 are extracted with stride 16. If multiple windows cover the same valid step, their reconstruction errors from

Equation (A.4.4) are averaged. The episode score is then

$$e^{\text{ep}} = \frac{1}{|\mathcal{T}_{\text{valid}}|} \sum_{t \in \mathcal{T}_{\text{valid}}} \bar{\rho}_t, \tag{A.5.1}$$

where $\bar{\rho}_t$ denotes the mean error across all windows covering step t.

A.5.2 Per-step score

To obtain a risk value at every recorded step, the implementation constructs a stride-one 32-step window ending at step t. When fewer than 32 observations are available, the missing history is left-padded with zeros. Let $\widetilde{X}_t \in \mathbb{R}^{32\times 21}$ denote this window. The raw score is

$$e_t = \frac{1}{32} \sum_{j=1}^{32} \rho_{t,j}, \tag{A.5.2}$$

where $\rho_{t,j}$ is the decoder reconstruction loss at position j of the window ending at t.

A.5.3 Calibration map

Three mappings from reconstruction error to risk are compared: percentile scaling, one-dimensional logistic regression, and ten-bin empirical calibration. Five-fold stratified out-of-fold predictions are used to estimate the Expected Calibration Error (ECE). Candidate calibrators are ranked by out-of-fold ECE and must additionally satisfy two implementation guards:

$$N_{\text{distinct}} \geq 20, \qquad \kappa_2 - \kappa_1 \geq 0.05, \qquad \kappa_3 - \kappa_2 \geq 0.05. \tag{A.5.3}$$

The selected map is logistic regression on the logarithmically transformed score:

$$r = \varphi(e) = \sigma\big(\alpha \log(1+e) + \delta\big), \qquad \sigma(v) = \frac{1}{1+e^{-v}}, \tag{A.5.4}$$

where α and δ are fitted parameters.

The calibration set contains 66 normal validation episodes and 66 attack episodes. Severity thresholds are the 50th, 95th, and 99th percentiles of the calibrated normal-validation risks. To avoid confusion with the trust variable τ_t, they are denoted by κ_j here:

$$(\kappa_1, \kappa_2, \kappa_3) = (0.2413,\ 0.4601,\ 0.5269). \tag{A.5.5}$$

A.6 Belief-State Historical Trust Filter

The implemented trust estimator is the belief-filtering component of the POMDP formulation in Chapter 5. The hidden state and observation sets are

$$\mathcal{S} = \{\text{safe}, \text{degraded}, \text{danger}, \text{deadend}\}, \tag{A.6.1}$$

$$\Omega = \{\text{nominal}, \text{caution}, \text{supervised}, \text{critical}\}. \tag{A.6.2}$$

The calibrated risk is discretised according to

$$o_t = \begin{cases} \text{nominal}, & 0 \le r_t < \kappa_1, \\ \text{caution}, & \kappa_1 \le r_t < \kappa_2, \\ \text{supervised}, & \kappa_2 \le r_t < \kappa_3, \\ \text{critical}, & \kappa_3 \le r_t \le 1. \end{cases} \tag{A.6.3}$$

Using the row order of $\mathcal{S}$ and the column order of Ω, the implemented observation matrix is

$$O = \begin{bmatrix} 0.3714 & 0.5105 & 0.0568 & 0.0613 \\ 0.2814 & 0.4701 & 0.0527 & 0.1957 \\ 0.1016 & 0.3894 & 0.0445 & 0.4645 \\ 0.0116 & 0.3490 & 0.0404 & 0.5989 \end{bmatrix}. \tag{A.6.4}$$

The safe and dead-end rows are constructed from the empirical false-positive and false-negative analysis, with Laplace smoothing 10^{-2}. The two intermediate rows are monotone interpolations:

$$O_{\text{degraded},:} = 0.75 O_{\text{safe},:} + 0.25 O_{\text{deadend},:}, \tag{A.6.5}$$

$$O_{\text{danger},:} = 0.25 O_{\text{safe},:} + 0.75 O_{\text{deadend},:}. \tag{A.6.6}$$

The action-independent transition matrix is

$$T = \begin{bmatrix} 0.970 & 0.025 & 0.005 & 0.000 \\ 0.100 & 0.700 & 0.180 & 0.020 \\ 0.050 & 0.100 & 0.550 & 0.300 \\ 0.000 & 0.000 & 0.030 & 0.970 \end{bmatrix}, \tag{A.6.7}$$

where rows represent the current state and columns the next state. The initial belief is

$$b_0 = \begin{bmatrix} 0.95 & 0.04 & 0.01 & 0 \end{bmatrix}^\top. \tag{A.6.8}$$

For observation o_t, the prediction and correction steps are

$$\bar{b}_t(s') = \sum_{s \in \mathcal{S}} T(s, s') b_{t-1}(s), \tag{A.6.9}$$

$$b_t(s') = \frac{O(s', o_t)\bar{b}_t(s')}{\sum_{\tilde{s} \in \mathcal{S}} O(\tilde{s}, o_t)\bar{b}_t(\tilde{s})}. \tag{A.6.10}$$

Historical trust is the linear projection

$$\tau_t = w^\top b_t, \qquad w = \begin{bmatrix} 1 & 0.67 & 0.33 & 0 \end{bmatrix}^\top. \tag{A.6.11}$$

No action-dependent transition, reward function, or POMDP policy is optimised in this implementation.

A.7 Role-Based Access-Decision Policies

A.7.1 Roles, sensitive steps, and permission gate

The controller chooses one role from

$$\mathcal{H} = \{\text{Full}, \text{Supervised}, \text{Prescribed}, \text{Quarantined}\}. \tag{A.7.1}$$

The corresponding integer levels are $L = 0, 1, 2, 3$.

For evaluation, an attempted action is considered sensitive when

$$q(a_t) \geq 0.35 \quad \text{and} \quad \nu_t \geq 0.30, \tag{A.7.2}$$

where ν_t is the de-standardised resource-sensitivity feature and

a_t	no action	navigate	invoke tool	escalate	observe	inspector
$q(a_t)$	0	0.30	0.85	0.40	0	0

(A.7.3)

The role-dependent permission mapping used during replay is given in Table A.7.1.

Table A.7.1: Role-dependent permission mapping used in offline replay.

Role	Implemented permission rule
Full	Allow every recorded action.
Supervised	Deny *invoke-tool* actions when resource sensitivity is at least 0.30; allow all other actions.
Prescribed	Deny *invoke-tool* and *escalate*; allow the remaining actions.
Quarantined	Allow only *observe* and no-action steps.

A.7.2 Shared suspicion score

The two optimisation-based policies use

$$s_t = \beta_r r_t + \beta_\tau (1 - \tau_t) + \beta_b b_t(\text{deadend}), \tag{A.7.4}$$

with

$$(\beta_r, \beta_\tau, \beta_b) = (2.0, 1.5, 0.0). \tag{A.7.5}$$

Thus the implemented score reduces to

$$s_t = 2r_t + 1.5(1 - \tau_t) \in [0, 3.5]. \tag{A.7.6}$$

A.7.3 Soft-threshold policy

The soft-threshold policy assigns

$$h_t = \begin{cases} \text{Full}, & s_t < 1.5, \\ \text{Supervised}, & 1.5 \le s_t < 2.5, \\ \text{Prescribed}, & 2.5 \le s_t < 3.5, \\ \text{Quarantined}, & s_t \ge 3.5. \end{cases} \tag{A.7.7}$$

After a new role is computed, a hold counter is set to five. The role is reused for the following five processed steps before a new decision is computed.

A.7.4 Discrete finite-horizon quadratic-cost controller

The controller labelled *QP-MPC* in the implementation is an enumerative discrete controller rather than a continuous quadratic program. For horizon $H = 5$, it evaluates every candidate role-level sequence

$$\mathbf{L} = (L_0, \dots, L_{H-1}) \in \{0, 1, 2, 3\}^H. \tag{A.7.8}$$

There are $4^5 = 1024$ candidate sequences.

The future risk and distrust signals are approximated by mean reversion:

$$\hat{r}_k = \gamma^k r_t + (1 - \gamma^k) r_{\text{base}}, \tag{A.7.9}$$

$$\hat{d}_k = \gamma^k (1 - \tau_t) + (1 - \gamma^k) d_{\text{base}}, \tag{A.7.10}$$

where

$$\gamma = 0.85, \qquad r_{\text{base}} = d_{\text{base}} = 0.10. \tag{A.7.11}$$

The predicted suspicion is

$$\hat{s}_k = 2\hat{r}_k + 1.5\hat{d}_k. \tag{A.7.12}$$

For role level L, define the permissiveness and restriction penalties

$$p(L) = \left(\frac{3 - L}{3}\right)^2, \qquad u(L) = \left(\frac{L}{3}\right)^2. \tag{A.7.13}$$

The evaluated cost is

$$J(\mathbf{L}) = \sum_{k=0}^{H-1} \left[\lambda_r \hat{s}_k p(L_k) + \lambda_u u(L_k)\right] \tag{A.7.14}$$

$$+ \lambda_s \left[\left(\frac{L_0 - L_{t-1}}{3}\right)^2 + \sum_{k=1}^{H-1} \left(\frac{L_k - L_{k-1}}{3}\right)^2\right]. \tag{A.7.15}$$

The parameters are

$$\lambda_r = 1, \qquad \lambda_s = 0.5, \qquad \lambda_u = 5.3344. \tag{A.7.16}$$

The utility weight λ_u is selected from a finite grid to approximately match the benign-action rate of the default soft-threshold policy. The first role of the minimum-cost sequence is applied and held using the same five-step counter as the soft-threshold policy.

A.7.5 Barrier-inspired threshold safety filter

The controller labelled *CBF-only* in the implementation is a barrier-inspired threshold filter. It recomputes the role at each valid step:

$$h_t = \begin{cases} \text{Quarantined}, & r_t > 0.95, \\ \text{Prescribed}, & r_t > 0.50, \\ \text{Prescribed}, & \tau_t < 0.30, \\ \text{Full}, & \text{otherwise}. \end{cases} \tag{A.7.17}$$

The critical-risk condition has precedence over the remaining conditions. No forward-invariance certificate is computed; consequently, this policy is interpreted as a simplified safety filter rather than a formal CBF controller.

A.8 Offline-Replay Protocol and Metrics

The access policies are evaluated on 444 normal episodes and 145 attack episodes, consisting of 41 successful, 72 partial, and 32 blocked attacks. The trajectories were originally collected under full privilege. During replay, all controllers receive the same precomputed sequences

$$\{r_t, \tau_t, b_t\}_{t=1}^{T}. \tag{A.8.1}$$

A selected role determines whether the recorded action would have been allowed, but the decision does not alter the following observation, action, risk, or trust value. The experiment therefore measures counterfactual permission decisions on fixed traces, not closed-loop behavioural adaptation.

The policy and trust filter are updated from the beginning of each episode, but controller metrics exclude steps with index smaller than the window length $W = 32$. This guard reduces the direct influence of incomplete history on the reported access-control metrics.

Let $\mathcal{J}_{\text{atk}}$ and $\mathcal{J}_{\text{norm}}$ denote the sets of sensitive steps in attack and normal episodes, respectively. Let $B_t = 1$ indicate that the selected role blocks the recorded action. The code reports the harm-prevention rate

$$\text{HPR} = \frac{\sum_{t \in \mathcal{J}_{\text{atk}}} B_t}{|\mathcal{J}_{\text{atk}}|}, \tag{A.8.2}$$

and the benign-action rate

$$\text{BAR} = 1 - \frac{\sum_{t \in \mathcal{J}_{\text{norm}}} B_t}{|\mathcal{J}_{\text{norm}}|}. \tag{A.8.3}$$

Here HPR should be interpreted literally as the blocking rate of sensitive actions occurring in attack-labelled episodes; the dataset does not provide a separate maliciousness label for every individual action.

The normal restriction rate is

$$R_{\text{norm}} = \frac{\#\{\text{scored normal steps assigned a non-Full role}\}}{\#\{\text{scored normal steps}\}}. \tag{A.8.4}$$

The analysis additionally records the first quarantine step and the proportion of

successful-attack episodes that reach quarantine.

A.9 Implementation-Specific Validity Considerations

The following points delimit the interpretation of the experimental results.

1. **Exploratory model selection.** Both latent priors are trained only on normal episodes, but the retained GMM variant is chosen after comparing ROC–AUC on the available attack corpus. The prior comparison is therefore exploratory rather than an untouched test-set selection.
2. **Supervised calibration.** The CVAE–LSTM representation is trained as a normal-only model, but the selected logistic calibration map uses 66 labelled attack episodes in addition to 66 normal validation episodes.
3. **Full-corpus downstream analysis.** The trust filter and access-control policies are subsequently applied to all 589 valid episodes. Their reported results are therefore retrospective full-corpus analyses rather than independent held-out generalisation tests.
4. **Aggregation transfer.** The calibrator is fitted using episode-level mean reconstruction scores and is then applied to stride-one window scores. Likewise, episode-level error statistics are reused to construct a per-step observation model. These are practical approximations necessitated by the available sample size.
5. **Window initialisation.** Before 32 observations are available, the stride-one risk windows are left-padded with zeros. The trust filter is updated during this initial period, although controller metrics exclude the first 32 steps.
6. **Stochastic latent sampling.** Reconstruction scores use a reparameterised latent sample rather than the posterior mean. Fixed random seeds make the stored run reproducible, but the score contains Monte Carlo variability.
7. **Offline and single-agent evaluation.** Controller actions do not affect later observations, and each episode is evaluated independently. Consequently, the experiment does not test intervention-induced behaviour changes or cross-agent resource coupling.