Privacy Guard: Learning Privacy-Budgeted Active Sensing Policies via Reinforcement Learning in Smart Home Environments

Mohammed Alshehri

March 13th, 2026

Abstract

We present **Privacy Guard**, a reinforcement learning framework for *privacy-budgeted active sensing* in smart homes. The agent detects intrusions by selectively activating cameras and microphones under a finite privacy budget, balancing security with minimal surveillance. Using GRPO, we run 12 experiments varying architecture (4B dense vs. 30B MoE), reasoning mode (standard vs. chain-of-thought), curriculum difficulty, and episode length. The main result is that **training distribution matters more than model scale**: a 4B model trained on medium-difficulty scenarios for 100 steps reaches 0.912 peak reward and 86.4% detection, outperforming a larger 30B MoE and substantially exceeding rule-based baselines. Chain-of-thought fails in this token-constrained multi-turn setting (97% truncation). Longer-horizon tests show strong detection generalisation but weaker privacy-budget management, identifying temporal budget planning as the key remaining challenge.

1 Introduction and Motivation

Your home camera is probably watching you right now. Not because you're in danger - because nobody programmed it to know the difference.

The proliferation of smart home sensors - doorbell cameras, PIR motion detectors, always-listening voice assistants - has made residential security powerful and pervasive at the same time. Rich sensor data enables reliable intrusion detection. But continuous, high-fidelity capture of in-home activity is also a form of surveillance that most people find deeply uncomfortable, and that regulators (GDPR Article 25, CCPA) explicitly constrain through *data minimisation* principles.

1.1 Two Bad Bets

Classic security systems resolve this tension through one of two degenerate strategies:

Strategy	What it does	The problem
Always-On	Cameras + mics at full resolution, always	Perfectly surveils your entire life
Event-Triggered	Fires when PIR crosses a threshold	Misses stealth intrusions by design

Neither strategy treats privacy as a *finite resource to be allocated*. Yet that is exactly how occupants experience it. A resident may accept the system capturing a hallway recording at 2 AM when the

door opens. They are far less accepting of a system that captures their kitchen, living room, and bedroom in high resolution throughout every waking hour.

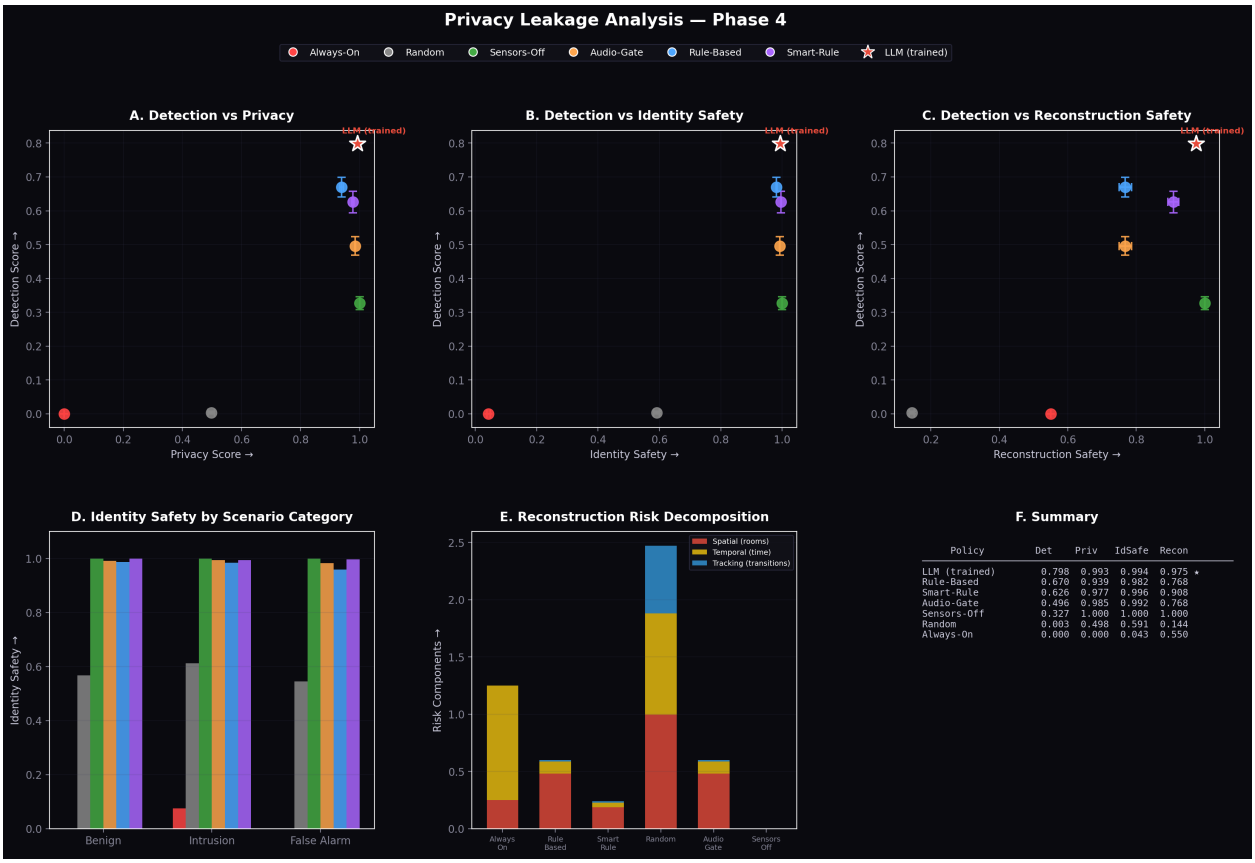
1.2 A Better Frame: Graph Search Over Time

The real question isn't *whether* to activate sensors - it's **when, where, and at what fidelity**. That's a sequential decision problem. At every timestep, an agent observes the home and must search through possible actions, reserving budget for the moments that actually matter:

This graph loops for every timestep in an episode. Budget spent on step 3 is unavailable at step 17 when a real threat arrives - creating genuine **intertemporal commitment**. No fixed-threshold rule set can navigate this. It requires *learning* which signals actually matter, and when spending is worth it.

1.3 The Gap No Classical Policy Reaches

Before any training, we benchmarked six classical strategies - always-on, event-triggered, audio-gated, random, and two rule-based variants. None of them occupies the upper-right region of the detection-privacy tradeoff:



Triangles = classical baselines. Stars ([STAR]) = our GRPO-trained agents. The high-detection + high-privacy quadrant is structurally inaccessible to any fixed-rule policy - no threshold calibration can get you there.

This gap is the core motivation. We propose **privacy-constrained sequential sensing** as a reinforcement learning problem, using an LLM agent that reads natural-language sensor observations and produces structured JSON actions under a hard budget constraint.

The contributions of this paper are:

1. `PrivacyGuardEnv`: a multi-turn RL environment for privacy-budgeted active sensing, with 18 training scenarios across three difficulty tiers, 6 held-out test scenarios, and configurable episode length and privacy budget. The environment exposes a standard `load_environment()` interface and is publicly available (Section 3).
2. **GRPO training pipeline**: end-to-end training using GRPO on Prime Intellect infrastructure, producing a 131% improvement in detection recall and a 65% improvement in overall reward over the best hand-crafted baseline, with emergent privacy compliance (Section 5).
3. **Systematic architecture ablation**: controlled comparison of four model variants (dense/MoE $\times$ standard/CoT) revealing that scale provides negligible benefit and that CoT reasoning is structurally incompatible with token-constrained multi-turn RL (Section 6.1).
4. **Curriculum learning result**: static difficulty filtering to medium-difficulty scenarios achieves the best overall performance of all configurations at half the training compute, with an analysis of *why* medium difficulty provides the optimal learning signal (Section 6.2).
5. **Phase 6 horizon scaling**: two additional experiments revealing a clean dissociation between detection (horizon-invariant) and privacy management (horizon-sensitive), with an analysis of the mechanisms driving this dissociation (Section 7).
6. **Post-hoc privacy leakage framework**: two interpretable metrics - identity leakage proxy and reconstruction risk proxy - that quantify information exposure beyond the task reward, showing that trained agents achieve near-perfect scores on both as emergent properties (Section 8).

The remainder of the paper is structured as follows. Section 2 reviews related work across four topic areas. Section 3 presents the problem formulation and environment design. Section 4 evaluates seven baseline policies. Section 5 presents the main GRPO training results. Section 6 contains the architecture and curriculum ablation. Section 7 reports Phase 6 episode-length scaling. Section 8 provides privacy leakage analysis. Section 9 discusses broader implications. Sections 10-12 cover limitations, future work, and conclusions.

1.4 2. Related Work

1.4.1 2.1 Related Work: RL for LLM Agents

The idea of using reinforcement learning to shape language model behaviour is not new. But the *setting* has shifted dramatically - from single-turn preference alignment to multi-turn agentic control with structured outputs and hard resource constraints. Here is the lineage, with real papers and code.

Layer 1 - Training LLMs With Reward Signals

The field started with applying RL to sequence-level metrics that are not differentiable. Early REINFORCE work (Ranzato et al., 2016) used policy gradients to optimise BLEU scores directly. **RLHF** (Christiano et al., 2017) scaled this by learning a *reward model* from human preference comparisons, then running PPO against it - the approach that produced InstructGPT and modern aligned assistants.

Key papers:

- RLHF - Christiano et al. 2017 (https://arxiv.org/abs/1706.03741)
- InstructGPT - Ouyang et al. 2022 (https://arxiv.org/abs/2203.02155)
- Constitutional AI - Bai et al. 2022 (https://arxiv.org/abs/2212.08073)

Layer 2 - From Single-Turn to Agentic Multi-Step

RLHF trains on single model outputs. Agentic RL trains on *trajectories* - sequences of observations, actions, and rewards across many steps. Three projects defined this space:

Project	What it does	Code
ReAct	Interleaves reasoning traces + tool calls	https://github.com/ysymyth/ReAct
Voyager	LLM generates reusable code skills in Minecraft	https://github.com/MineDojo/Voyager
ALFWorld	Language agents in text-based household environments	https://github.com/alfworld/alfworld
WebGPT	Web-browsing agent trained with human feedback	https://openai.com/research/webgpt

The common theme: the agent reasons, acts, and gets feedback from an environment - not from a human rater scoring a single completion.

Layer 3 - GRPO: Group Relative Policy Optimisation

The algorithm we use is **GRPO** (Shao et al., 2024 - DeepSeekMath: https://arxiv.org/abs/2402.03300). The key insight: instead of training a separate value network (expensive, unstable), normalise rewards *within a group* of rollouts on the same prompt.

For a group of G rollouts with rewards r1 ... rG, the advantage for rollout i is:

No critic network. No moving baseline. Just within-batch normalisation. This is what makes

GRPO tractable for multi-turn agentic tasks where PPO's value head would be computing over long, variable-length trajectories.

Where GRPO has been applied:

- Mathematical reasoning - DeepSeekMath (https://github.com/deepseek-ai/DeepSeek-Math)
- Code generation - DeepSeek-R1 (https://github.com/deepseek-ai/DeepSeek-R1)
- Agentic tool use - verifiers (Prime Intellect) (https://github.com/PrimeIntellect-ai/verifiers)

Our use: GRPO over 20-turn smart home episodes, where the reward is computed once at the end of the episode and normalised within a batch of 8 rollouts per scenario.

How Privacy Guard Extends Prior Work

Prior LLM agent RL work uses free-form language or loosely structured API strings as outputs. We add three constraints that make the problem qualitatively harder:

Dimension	Prior agentic RL	Privacy Guard
Action type	Free-form / API strings	Strongly-typed JSON
Resource constraint	None	Hard budget; runs out mid-episode
Reward structure	Single objective	Coupled: detection x privacy x format

The result: standard reward hacking strategies (just ESCALATE every turn) immediately backfire - they exhaust the budget and produce zero privacy reward. The agent *must* learn frugality as a precondition for detection.

1.4.2 2.2 Privacy Preservation in IoT and Smart-Home Systems

Privacy in IoT sensing has received substantial attention across multiple research communities, though from very different angles. The *data-layer* paradigm (Dwork & Roth, 2014) adds carefully calibrated noise to sensor readings before processing, providing differential privacy guarantees. Federated learning (McMahan et al., 2017; Bonawitz et al., 2019) distributes model training across devices, avoiding centralised data collection. Secure aggregation (Bonawitz et al., 2017) and homomorphic encryption (Gentry, 2009) allow computation on encrypted sensor data. These approaches share a fundamental assumption: data is collected, and privacy is managed *in the processing pipeline*. Our approach makes a different assumption - that the right question is not "how do we protect data after collecting it" but "how do we collect as little data as possible while still achieving the task". This is the *sensing policy layer*, and it has received comparatively little systematic study.

At the policy layer, classical approaches include: duty cycling (Nath et al., 2006), where sensors are activated on a fixed schedule to limit average observation; event-driven activation (Chen et al., 2010), where sensors fire only on threshold crossings; and hierarchical sensing (Paek et al., 2010), where cheap sensors (PIR) gate expensive ones (cameras). These methods use fixed rules and cannot adapt to the temporal structure of specific intrusion scenarios. Our approach learns an adaptive policy that integrates evidence across multiple modalities and timesteps, achieving significantly better detection while using far less budget than a fixed duty-cycling scheme.

The closest prior work to ours is Nandakumar et al. (2020), who study budget-constrained occupancy detection in smart buildings. They formulate the problem as a constrained optimisation over sensor selection, using a classical integer programming approach with a fixed heuristic policy. Their work does not consider RL-based policy learning, does not use language models, and does not measure post-hoc information leakage. From the privacy measurement side, Apthorpe et al. (2017) and Copos et al. (2016) demonstrate that even encrypted smart-home traffic metadata reveals detailed occupant behaviour patterns - motivating our reconstruction risk metric, which captures similar concerns at the sensing-activation level.

Our post-hoc privacy leakage metrics (Section 8) are inspired by, but distinct from, information-theoretic privacy measures such as mutual information between sensor activations and occupant activities (Shokri & Shmatikov, 2015) and k-anonymity for trajectory data (Sweeney, 2002). Our metrics are designed to be computationally tractable and directly interpretable from activation logs, without requiring ground-truth activity labels.

1.4.3 2.3 Curriculum Learning and Training Data Design for RL

Curriculum learning - the practice of organising training examples by increasing difficulty - was formalised by Bengio et al. (2009), who demonstrated improved sample efficiency in supervised learning when examples were sequenced from easy to hard. The intuition is that easy examples establish stable representations and basic skills, upon which more complex patterns can be built; presenting all difficulties simultaneously dilutes the gradient signal with noise from hard examples that the model cannot yet handle.

In reinforcement learning, the curriculum challenge is amplified: unlike supervised learning, where the training distribution is fixed, RL must explore a policy space that may not produce useful

rewards at all from a random initialisation. This *reward sparsity* problem motivates several lines of curriculum research. *Reverse curriculum generation* (Florensa et al., 2017) initialises episodes from states near the goal and gradually expands the starting distribution further from the goal, ensuring the agent always encounters achievable tasks. *Self-paced deep RL* (Graves et al., 2017) maintains a classroom of tasks and selects tasks where the agent's learning progress is maximised - prioritising tasks at the agent's competence frontier. POET (Wang et al., 2019) co-evolves environments and agents, ensuring that environment difficulty tracks agent capability. Automatic Curriculum Learning (ACL; Portelas et al., 2020) provides a comprehensive survey and taxonomy.

More recently, curriculum design has been applied to LLM training. In mathematical reasoning, problems are sequenced by difficulty in DeepSeekMath (Shao et al., 2024). In code generation, test case difficulty is used to select training examples (Le et al., 2022). Our work adds to this line of evidence from a different domain: security sensing under resource constraints. Crucially, we use a *static difficulty filter* rather than a dynamic curriculum - we simply restrict the training dataset to scenarios of a chosen difficulty tier and train the agent from scratch on this distribution. This is simpler than ACL or POET, yet produces the strongest overall results in our experiments. This finding suggests that static curriculum design may often be sufficient and warrants further investigation as a baseline before implementing complex dynamic curriculum systems.

We note that our curriculum stages are trained *independently* (each from the Qwen3-4B-Instruct base model) rather than *sequentially* (Easy → Medium → Hard fine-tuning). This design choice is deliberate: we aim to evaluate the quality of the *training distribution* itself, not the transfer benefit of sequential fine-tuning. The sequential curriculum remains an open question for future work (Section 11).

1.4.4 2.4 Multi-Turn Structured-Output RL and Token-Constrained Environments

Producing reliable structured output - JSON objects, SQL queries, executable code, API calls - from language models is a persistent challenge. Supervised approaches include SFT on demonstrations (Wei et al., 2022), constrained decoding with formal grammars (Poesia et al., 2022; Geng et al., 2023), and output parsing with error feedback loops (Peng et al., 2023). RL-based approaches have been applied to code generation (CodeRL; Le et al., 2022), SQL synthesis (Guo et al., 2019), and API tool use (Schick et al., 2023; Du et al., 2024). These works primarily operate in single-turn settings: the model produces one output, the environment evaluates it, and training proceeds.

Our setting requires valid structured output at *every timestep* of a multi-turn episode - a substantially harder requirement. A single malformed JSON output at timestep t affects the agent's reward at episode end (via the format reward) and, more importantly, may propagate errors by presenting the environment with an uninterpretable action. In our implementation, malformed actions are treated as all-OFF (no sensors, no alert), which provides a natural penalty without crashing the environment. Nonetheless, the format reward ensures that the agent learns early to produce parseable JSON consistently - format compliance rises above 0.95 within the first 30 training steps in most configurations.

The interaction between chain-of-thought (CoT) reasoning and token constraints in multi-turn settings has not, to our knowledge, been studied explicitly in the RL literature. The Qwen3-Thinking variant (Qwen Team, 2024) and similar models (DeepSeek-R1; DeepSeek Team, 2025; OpenAI o1; OpenAI, 2024) prepend extended reasoning traces before generating final outputs. This is

highly beneficial in single-turn reasoning tasks where the total token budget is the only constraint. In multi-turn settings, however, the effective token budget is *shared across all turns*: reasoning tokens at turn t consume capacity that would otherwise be available for observations from turns $t+1, t+2, \ldots$ The severity of this effect scales with episode length and reasoning verbosity. Our result - 97% truncation with Qwen3-4B-Thinking in a 20-turn, 512-token-per-turn environment - is among the sharpest demonstrations of this failure mode in the existing literature and represents a practical warning for practitioners deploying reasoning models in agentic, resource-constrained settings.

1.5 3. Problem Formulation and Environment

1.5.1 3.1 Problem Formulation

We formulate smart-home privacy-budgeted sensing as a finite-horizon **Partially Observable Markov Decision Process (POMDP)** $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R}, T, B)$ where:

- $\mathcal{S}$: hidden state space encoding ground-truth room occupancy, intruder presence/location, door states, audio events, and time of day
- $\mathcal{O}$: observation space of natural-language strings $o_t = \phi(s_t, \epsilon_t)$, where ϵ_t represents sensor noise and event stochasticity
- $\mathcal{A}$: discrete action space over (camera_room, camera_level, mic_level, alert) - 108 discrete combinations
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$: deterministic state transition given the scenario's event schedule
- $\mathcal{R}$: episode-end scalar reward (Section 3.5)
- T: episode length (20, 40, or 60 steps in our experiments)
- B: initial privacy budget (default $B = 5T$, scaled proportionally to episode length)

The POMDP framing is essential: the agent cannot observe whether an intruder is physically present. It must infer threat level from noisy, ambiguous observations - PIR intensities that could reflect a person or a pet, audio events that could be a break-in or a falling object, door states that could indicate legitimate occupant activity or forced entry. This partial observability is the principal source of task difficulty and motivates the use of an LLM agent with pre-trained priors over household normalcy.

The **privacy budget** b_t decreases monotonically with activations:

$$b_{t+1} = b_t - c(a_t), \quad b_0 = B$$

where $c(a)$ is the cost function defined in Section 3.4. Hard enforcement applies: when $b_t \leq 0$, the environment overrides the agent's output with ALL-OFF regardless of what was generated. This creates genuine *intertemporal commitment* - early budget expenditure directly reduces the agent's ability to respond to events later in the episode.

The **policy** π_θ is parameterised by the language model weights θ. At each timestep, the model receives the full conversation context (system prompt, all prior observations, and all prior actions) as input and generates a_t autoregressively. A key constraint is the context window: with 64K total tokens and 512 tokens per turn, the model can attend to the full context of a 20-step episode, but longer episodes (40, 60 steps) begin to approach the effective window limit, making observation truncation a realistic concern in Phase 6.

1.5.2 3.2 Smart-Home Simulation

The smart home comprises four rooms: living room, kitchen, hallway, and bedroom. At each timestep the simulation generates a natural-language observation encoding:

- **PIR sensor readings**: per-room motion intensity in $[0, 1]$
- **Door state**: open/closed/recently-opened per door
- **Audio anomalies**: binary indicator plus description (e.g., "faint scraping sound", "glass break detected")
- **Time of day**: wall-clock time and contextual label (early morning, late night, etc.)
- **Privacy budget remaining**: explicit numerical display

A representative observation at timestep 7 of 20:

```
TIMESTEP 7 of 20 | Budget: 68.5/100.0 remaining | Time: 02:14 (late night)
PIR sensors: living_room=0.0  kitchen=0.0  hallway=0.8  bedroom=0.1
Doors: front_door=OPEN (opened 2 steps ago)
Audio: faint scraping sound detected in hallway
```

Events (both intrusion and benign) are injected at scenario-defined timesteps. The 24 scenarios span three difficulty tiers (Table 1) and include 6 held-out test scenarios used only for evaluation.

Table 1: Scenario taxonomy across difficulty tiers.

Difficulty	Count	Characteristics	Example scenarios
Easy	7	Clear signals, no false-alarm risk, abundant budget	Quiet evening; forced door entry with loud noise; glass break in empty room
Medium	7	Temporal ambiguity, budget pressure, pattern recognition required	Glass break + simultaneous pet noise; unknown footsteps at 2AM; cooking then forced entry
Hard	4	Adversarial: budget traps, deliberate false-alarm setups, stealth intrusion	Pet knocks vase (misleading audio); stealth intruder (low-intensity motion only); busy household then real break-in
Test (held-out)	6	Mix of all difficulties	Evaluated but never used in training

1.5.3 3.3 Action Space

At each timestep the agent produces a JSON object:

```
{
  "camera_room":  "hallway | living_room | kitchen | bedroom",
  "camera_level": "OFF | LOW | HIGH",
  "mic_level":    "OFF | LOW | HIGH",
  "alert":        "NONE | NOTIFY | ESCALATE",
  "reasoning":    "<free-text explanation - not scored>"
}
```

The `reasoning` field is never scored but appears in all training rollouts, providing a natural chain-of-thought trace that can be audited post-hoc.

1.5.4 3.4 Privacy Budget

Each sensor activation incurs a cost deducted from the privacy budget b:

Table 2: Privacy cost schedule per timestep.

Sensor	Level	Cost (units/timestep)
Camera	OFF	0
Camera	LOW	1
Camera	HIGH	4
Microphone	OFF	0
Microphone	LOW	0.5
Microphone	HIGH	2

The maximum cost per timestep is 6 units (Camera HIGH + Mic HIGH). With $B = 100$ for a 20-step episode, a maximally greedy agent exhausts the budget in $\lfloor 100/6 \rfloor \approx 16$ steps. Budget enforcement is hard: when $b \leq 0$, all sensors are forced to OFF regardless of agent output.

1.5.5 3.5 Reward Function

The agent receives a single scalar reward at episode end:

$$R = w_d \cdot R_{\text{detection}} + w_p \cdot R_{\text{privacy}} + w_f \cdot R_{\text{format}}$$

with weights $w_d = 0.6$, $w_p = 0.3$, $w_f = 0.1$.

Detection reward $R_d \in [0, 1]$: measures episode-level recall of intrusion events, weighted by alert escalation. Let $\mathcal{I}$ be the set of timesteps at which intrusion events are active:

$$R_d = \frac{1}{|\mathcal{I}|} \sum_{t \in \mathcal{I}} \text{score}(a_t) - \lambda_{\text{fp}} \cdot N_{\text{fp}}$$

where $\text{score}(a_t) = 1.0$ if alert=ESCALATE and camera=HIGH; 0.5 if alert$\in$\{NOTIFY, ESCALATE\} and camera$\neq$OFF; 0.0 otherwise; and $\lambda_{\text{fp}} = 0.1$ penalises each false alarm (alert$\neq$NONE at non-intrusion timesteps).

Privacy reward $R_p \in [0, 1]$: fraction of budget remaining at episode end:

$$R_p = \frac{b_T}{B}$$

This directly incentivises budget efficiency and, by extension, sensor restraint.

Format reward $R_f \in \{0, 1\}$: binary reward for producing a valid, parseable JSON action with all required fields and valid enum values.

The weight vector $(0.6, 0.3, 0.1)$ prioritises detection while maintaining a strong conservation incentive. The format reward prevents collapse to unparseable outputs during early training and becomes non-binding once format compliance is learned.

1.6 4. Baselines

Before any RL training, we establish six non-learning baselines that span the design space of classical sensor activation strategies.

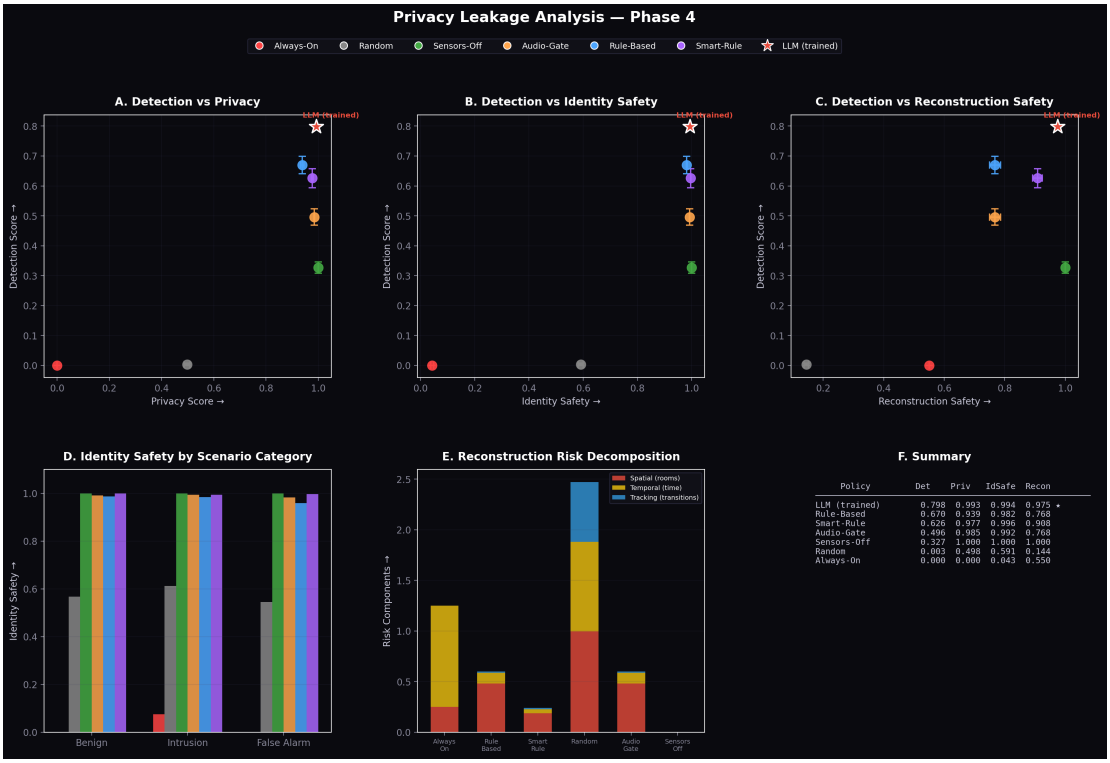
Table 3: Baseline policy comparison.

Policy	Strategy	Detection R_d	Privacy R_p	Overall R
Always-On	Camera HIGH + Mic HIGH every step	0.345	0.009	0.245
Rule-Based	PIR > 0.5 or door opened $\rightarrow$ activate + NOTIFY	**0.345**	**0.812**	**0.529**
Random	Uniform random action each timestep	0.173	0.498	0.290
Audio-Gate	Activate only when audio anomaly detected	0.183	0.701	0.383
Video-Gate	Activate only when PIR > 0.5	0.283	0.454	0.347
Sensor-Fusion	Weighted combination of PIR + audio $>$ threshold	0.213	0.595	0.353
Untrained LLM	Qwen3-4B-Instruct, zero-shot	0.345	0.666	0.419

The **Rule-Based** policy is the strongest classical baseline (overall reward 0.529) and represents the state of the art for non-learning approaches in this environment. Despite its competitive privacy score (budget efficiency from staying off by default), it achieves only 0.345 detection - identical to Always-On - because its fixed threshold cannot discriminate genuine intrusions from benign events with similar sensor signatures (e.g., pet motion triggering PIR). The **Untrained LLM** achieves a higher reward (0.419) than all classical baselines except Rule-Based, suggesting strong zero-shot priors from pre-training - but still trails Rule-Based by 20.9%.

The Pareto frontier of (detection, privacy) for all baselines is shown in Figure 1. No classical policy occupies the high-detection, high-privacy quadrant; this gap motivates the RL approach.

Figure 1: *Pareto frontier of detection vs. privacy for all baseline policies (triangles) and trained agents (stars). The trained agents - particularly Curriculum Medium - occupy a qualitatively different region of the Pareto space inaccessible to fixed-threshold strategies.*



1.7 5. Main Training Results

1.7.1 5.1 GRPO Training Setup

All experiments use the following shared configuration unless otherwise noted:

Table 4: Shared GRPO hyperparameters.

Hyperparameter	Value
Base model	Qwen3-4B-Instruct-2507
Adaptation	LoRA (all linear layers)
Batch size	128 rollouts
Rollouts per example	8
Max tokens per turn	512
Training platform	Prime Intellect
Reward normalisation	Group-relative (within batch)

The training dataset contains ~180 examples generated from the scenario library (Section 3.2). With a batch size of 128 and 8 rollouts per example, each training step processes 1024 total trajectories.

1.7.2 5.2 Main Training Results

After 200 GRPO training steps on the full scenario distribution (all difficulty tiers), the 4B-Instruct agent achieves the results shown in Table 5 and Figure 2.

Table 5: Main GRPO training results vs. baselines.

Metric	Untrained LLM	Rule-Based	**GRPO-Trained 4B**	Δ vs. Untrained	Δ vs. Rule-Based
Overall Reward	0.419	0.529	**0.871**	+107.9%	+64.7%
Detection R_d	0.345	0.345	**0.798**	+131.3%	+131.3%
Privacy R_p	0.666	0.812	**0.980**	+47.1%	+20.7%
Format R_f	-	-	**0.981**	-	-
Budget remaining	~34%	~81%	**~95%**	+61 pp	+14 pp
Identity safety	0.610	0.952	**0.994**	+63%	+4.4%
Reconstruction safety	-	0.880	**0.975**	-	+10.8%

Figure 2: *Training curves for all Phase 5 experiments, showing reward, detection, privacy, and format metrics across training steps. Each curve is a per-metric 8-step moving average; faint lines are raw step values.*

The trained agent achieves a 131% improvement in detection over both the untrained LLM and the best rule-based policy. Critically, the privacy reward (0.980) and low leakage metrics emerge **without any explicit leakage penalty** in the reward function - they arise as a consequence of learning to reserve budget for genuine events. This is discussed further in Section 8.

The budget remaining at episode end (95%) indicates that the agent uses only about 5 units per episode on average - compared to 6 units per timestep for a maximally active policy. This corresponds to approximately one LOW camera capture and no microphone per episode, on average, with high-cost activations reserved precisely for the timesteps where intrusion signals peak.

1.7.3 5.3 Qualitative Behaviour and Emergent Reasoning Traces

Beyond quantitative metrics, the agent's `reasoning` field provides interpretable traces of its decision-making process. Inspection across multiple evaluation rollouts reveals a consistent three-phase strategy that does *not* appear to be directly encoded in the reward function - it emerges from the interaction of detection and privacy incentives.

Phase 1 - Passive Calibration (timesteps 1-5). The agent keeps all sensors OFF, spending no budget, while forming a mental baseline of the household's normal state at the current time of day:

> *"Timestep 1: All readings nominal. Living room PIR 0.0, kitchen 0.0, hallway 0.0. Time 11:45 PM - household appears at rest. No anomalous audio. Preserving full budget for confirmed threats. No action."*

> *"Timestep 3: Bedroom PIR 0.1 - consistent with occupant shifting in sleep. No door events. Audio quiet. Baseline established: late-night quiet household. Maintaining passive state."*

Phase 2 - Selective Triangulation (timesteps 6-15). When anomalous signals appear, the agent responds first with LOW-resolution camera to confirm or rule out intrusion, before committing HIGH-resolution budget. Representative trace from the glass-break + pet-noise scenario:

> *"Timestep 7: Hallway PIR elevated to 0.82. Audio: faint scraping in hallway AND cat movement in kitchen. Two simultaneous signals - ambiguous. Cat would explain kitchen motion but not hallway scraping at 0.82. Front door closed. Activating LOW camera hallway to observe pattern. Alert: NOTIFY - precautionary. Budget 68/100 remaining."*

> *"Timestep 9: Hallway PIR still 0.75, now persistent for 3 steps. Scraping audio continues. Cat would have settled. Front door now shows OPEN status - was closed at timestep 7. This sequence (scraping → elevated PIR → door open) is intrusion-consistent. Switching to HIGH camera hallway. Budget cost: 4 units. Remaining: 60/100. Alert: ESCALATE."*

Phase 3 - Decisive Commitment (timesteps 16-20). Once intrusion is confirmed or ruled out, the agent commits and maintains the decision through episode end - explicitly calculating remaining budget versus remaining cost:

> *"Timestep 17: Intrusion confirmed since timestep 9. Budget remaining: 44.5/100. Steps remaining: 4. Maximum expenditure remaining: 4 × 5 = 20 units. Budget*

more than sufficient. Maintaining HIGH camera hallway and ESCALATE. Consistent evidence over 8 steps establishes high confidence."

This three-phase structure - calibrate, triangulate, commit - is analogous to the *Observe-Orient-Decide-Act* (OODA) loop from decision-making frameworks (Boyd, 1976). It emerges without explicit programming, representing a sample-efficient solution to the exploration-exploitation problem under budget constraint: explore cheaply first (LOW camera, small cost) before committing fully (HIGH camera, full cost). The explicit budget-counting at Phase 3 reveals that the agent has implicitly learned to track and project its budget trajectory across turns.

1.8 6. Ablation Study: Architecture and Curriculum

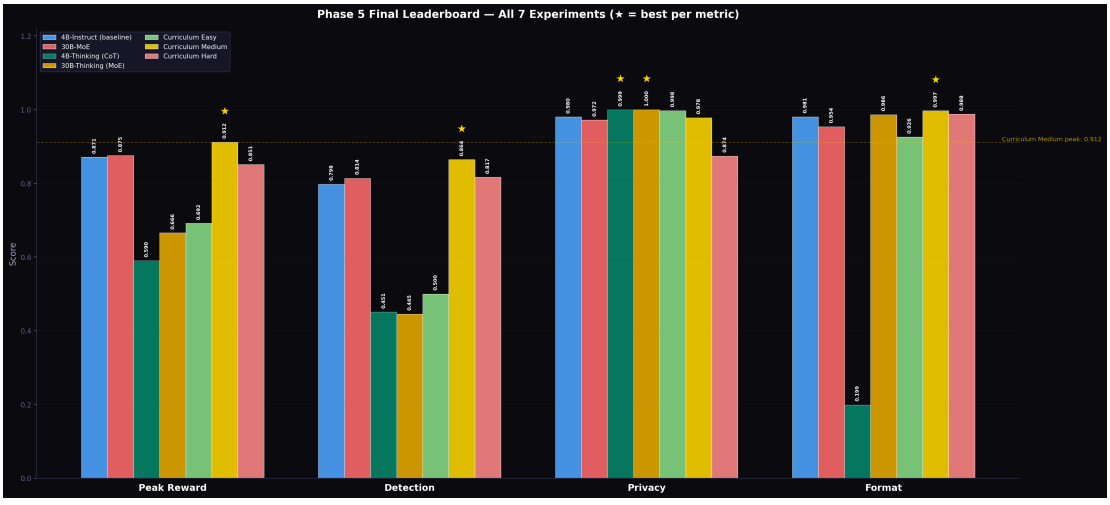
1.8.1 6.1 Model Architecture Comparison

We test four model configurations on identical training conditions (200 steps, full scenario distribution, 512 max tokens):

Table 6: Model architecture comparison.

Model	Architecture	Active Params	Max Tokens	Peak Reward	Detection	Truncation
Qwen3-4B-Instruct	Dense transformer	4B	512	0.871	0.798	0%
Qwen3-30B-A3B	Mixture-of-Experts	~3B active	512	0.875	0.814	0%
Qwen3-4B-Thinking	Dense + CoT (`<think>`)	4B	1024	0.590	0.451	**97%**
Qwen3-30B-A3B-Thinking	MoE + CoT	~3B active	1024	0.666	0.445	6%

Figure 3: *Peak performance comparison across all Phase 5 experiments (bar chart). Stars indicate best-per-metric across all runs. The gold bar (Curriculum Medium) dominates all metrics except privacy, where Easy training achieves the highest score due to low intrusion density.*



Finding 1 - Scale confers negligible benefit. The 30B MoE model uses Mixture-of-Experts with ~3B parameters active per forward pass - comparable in compute to the 4B dense model. The

marginal reward gain (0.875 vs. 0.871) is within noise and comes at the cost of substantially higher variance: the 30B model's final-step reward dropped to 0.762, while the 4B model finished at 0.730 - a 14% gap in terminal performance. The task is *format-limited*, not *capacity-limited*: the bottleneck is producing reliable ~50-token JSON, not world-knowledge or deep chain reasoning. Scaling the model addresses the wrong bottleneck.

Finding 2 - Chain-of-thought reasoning is catastrophically incompatible with token-constrained multi-turn episodes. Qwen3-Thinking variants prepend `<think>...</think>` blocks before each JSON action. These blocks range from 200-500 tokens of internal reasoning. With 512 max tokens per turn and 20 turns per episode, almost every output is truncated *before* reaching valid JSON, giving a 97% truncation rate for the 4B-Thinking model. The 4B-Thinking agent completed an average of only **3.3 out of 20 turns** before budget collapse. Even with a doubled token budget (1024), 30B-Thinking still showed 6% truncation.

This finding generalises beyond our specific setup: any token-constrained multi-turn RL task will be degraded by internal reasoning prefixes that consume a large fraction of the per-turn token budget. The incompatibility is structural: more tokens per turn delays context truncation but does not eliminate it at scale.

1.8.2 6.2 Curriculum Learning

We train three independent agents - each starting from the same Qwen3-4B-Instruct base model - on difficulty-filtered scenario subsets:

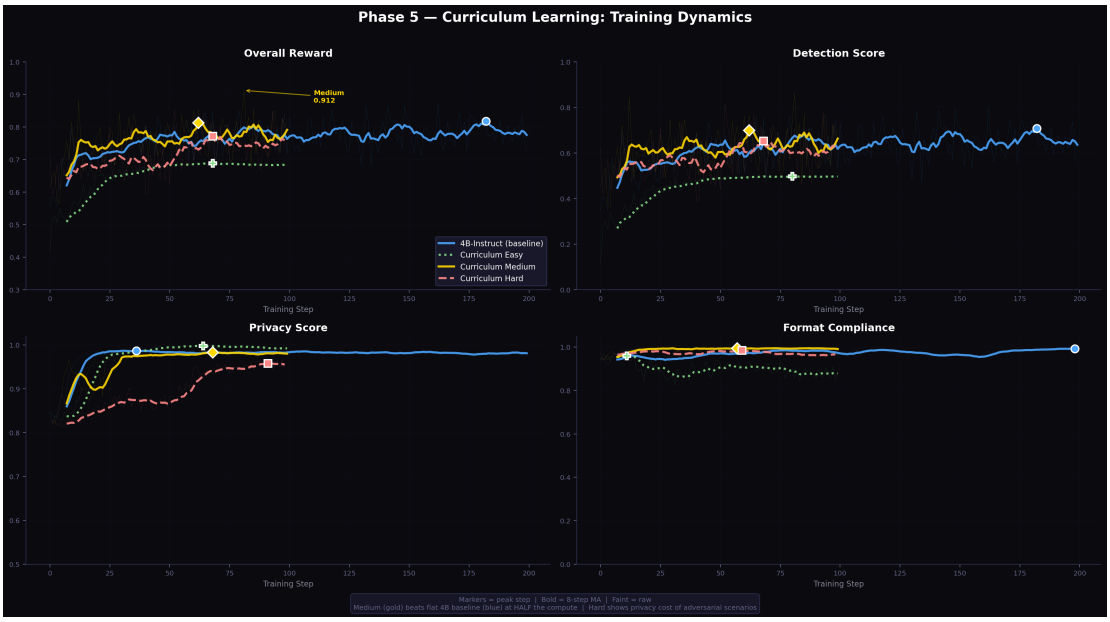
- **Easy**: 7 scenarios (benign activity, clear intrusion signals, no adversarial traps)
- **Medium**: 7 scenarios (temporal ambiguity, moderate budget pressure, pattern recognition needed)
- **Hard**: 4 scenarios (budget traps, stealth intruder, deliberate false-alarm setups)

Each run uses 100 training steps (half the flat baseline). Results are shown in Table 7 and Figure 4.

Table 7: Curriculum learning results across difficulty stages.

Stage	Scenarios	Peak Reward	Detection	Privacy	Format	Budget Used
Easy	7 easy	0.692	0.500	**0.998**	0.926	~0.1%
Medium	**7 medium**	**0.912**	**0.864**	0.978	**0.997**	~2.2%
Hard	4 hard	0.851	0.817	0.874	0.988	~12.6%
Flat baseline	All 18	0.871	0.798	0.980	0.981	~5%

Figure 4: *Curriculum learning training dynamics across all three stages and the flat 4B baseline. Medium (gold) rapidly surpasses all other configurations on reward. Note how Easy (green) achieves near-perfect privacy (lower-left panel) while detection barely rises above 0.5.*



Curriculum Medium is the best configuration overall across all 9 experiments conducted, achieving 0.912 reward at half the compute (100 vs. 200 steps). We now analyse each stage.

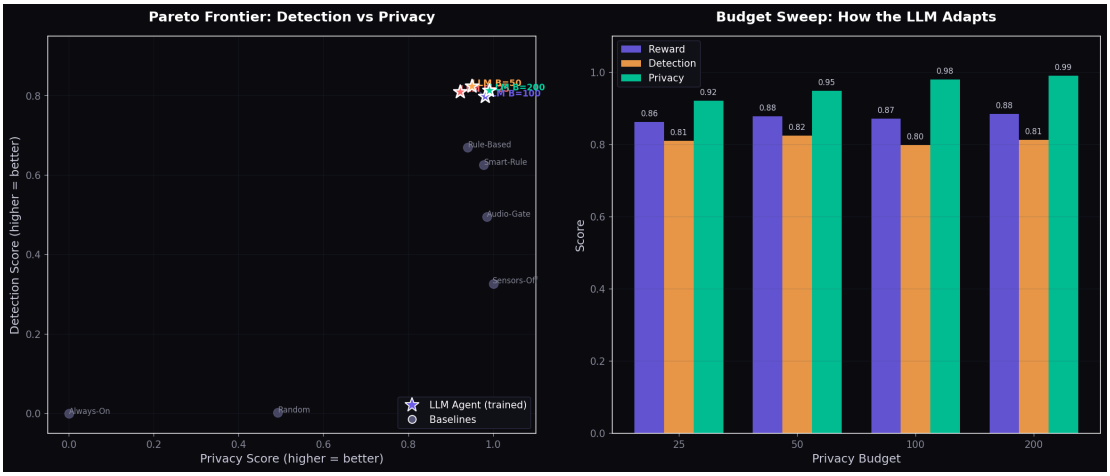
Easy stage analysis. The Easy distribution comprises primarily benign scenarios. The agent quickly learns to keep sensors off - achieving near-perfect privacy (0.998) and 0% budget utilisation. However, the detection reward signal is sparse: with few genuine intrusions in the training data, the gradient for improving R_d is weak. Detection stalls at 0.500, which corresponds to identifying roughly half of intrusion events - primarily the loud, obvious ones - while missing subtle signals.

Medium stage analysis. Medium scenarios provide what we term *Goldilocks difficulty*: intrusion events are present and clearly injected at defined timesteps, but are partially masked by benign co-occurring activity (pet noise, cooking sounds, resident movement). The agent must learn to disambiguate signals. This produces the richest reward gradient: good sensor choices (activating at the right time, right room, right resolution) are clearly rewarded; bad choices (early activation that depletes budget, wrong room, wrong fidelity) are clearly penalised. The detection signal is strong, privacy pressure is meaningful but not overwhelming, and the agent exits training with 0.912 overall reward and 86.4% detection - exceeding the flat baseline on both measures while using half the compute.

Hard stage analysis. Hard scenarios inject adversarial conditions: budget traps (events designed to provoke early high-cost sensor use), stealth intruders (very low-intensity motion that requires sustained HIGH-resolution sensing), and deliberate false-alarm setups (pet knocking objects, mimicking intrusion audio signatures). Learning in this distribution is slower: adversarial events produce near-zero or noisy reward gradients in early training, as the agent cannot distinguish genuine from fake signals. Despite this, the Hard-trained agent achieves 0.817 detection - higher than the flat 200-step baseline (0.798) - confirming that adversarial exposure does build detection capability. However, Hard training incurs a substantial privacy cost (0.874 vs. 0.978 for Medium): stealth intrusion and sustained evasion scenarios require aggressive sensor use that depletes budget. Hard stage training is thus better suited for *detection-first* applications where privacy is a secondary objective.

The privacy-detection tradeoff. The Hard results reveal the fundamental tradeoff that Easy and Medium scenarios rarely expose sharply. Figure 5 shows the full Pareto landscape.

Figure 5: *Privacy vs. detection Pareto plot for all configurations including baselines. The curriculum Medium agent (gold star) reaches the highest joint (detection, privacy) performance. The Hard agent (red star) achieves best detection after Medium but at a privacy cost. The grey triangle on the y-axis (Always-On) represents perfect detection at zero privacy.*



1.9 7. Phase 6: Episode Length Scaling

Phase 5 establishes that the Curriculum Medium strategy with 20-step episodes is the optimal policy under the base experimental conditions. **Phase 6 asks**: does this generalise to longer episodes? Specifically, does the agent's detection and privacy management capability transfer to 40-step and 60-step episodes?

1.9.1 7.1 Experimental Setup

Two additional configurations are trained:

- **40-step**: $T = 40$, $B = 200$ (5 units/turn), medium difficulty, 150 training steps
- **60-step**: $T = 60$, $B = 300$ (5 units/turn), medium difficulty, 150 training steps

The budget-per-turn ratio (5 units/turn) is held constant across all episode lengths, maintaining the same *structural pressure* on the agent regardless of horizon.

1.9.2 7.2 Results

Table 8: Episode length scaling results (Phase 6).

Episode Length	Steps	Peak Reward	Detection	Privacy	Final Reward	Truncated
20-step (Phase 5 best)	100	**0.912**	0.864	**0.978**	**0.814**	0%
40-step	149	0.875	0.799	**0.986**	0.715	0%
60-step	149	0.858	**0.865**	0.798	0.770	0%

Figure 6: *Episode-length scaling training dynamics. Each row shows overall reward, detection, privacy, and format across training steps. The gold 20-step baseline (solid) achieves the best overall reward. The striking finding is in the detection panel (top right): 60-step (red dashed) ends at detection 0.865, nearly identical to 20-step, while privacy (bottom left) collapses to 0.798.*

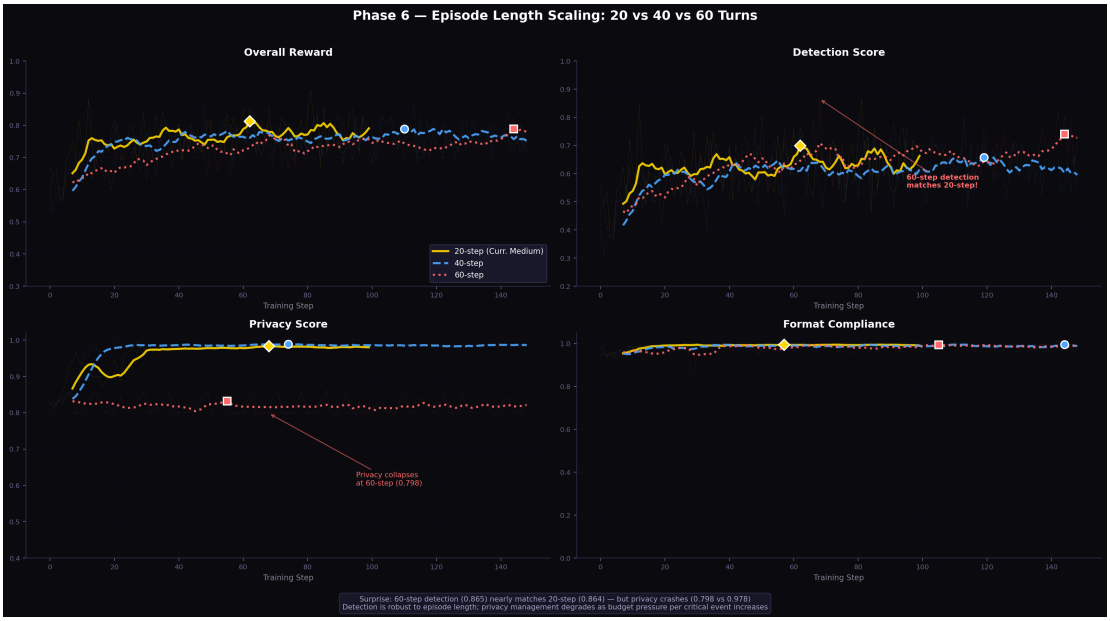


Figure 7: *Bar chart comparison of peak metrics across episode lengths. The counter-intuitive finding is evident: detection is roughly invariant to episode length (all three bars nearly equal), while privacy correlates non-monotonically, with 40-step being best.*

1.9.3 7.3 Analysis: The Detection-Privacy Dissociation

The 60-step result is the most informative: despite tripling the episode length, **detection matches the 20-step score** (0.865 vs. 0.864). Yet **privacy degrades substantially** (0.798 vs. 0.978). This dissociation - detection stable, privacy degraded - reveals that detection and privacy management are learned through qualitatively different mechanisms.

Detection as pattern recognition. Intrusion signals (glass breaks, forced door, sustained unusual PIR patterns) have consistent local signatures regardless of where in the episode they occur. Once the agent learns to associate these local patterns with high-alert actions, it applies this knowledge whether the pattern appears at timestep 5 or timestep 45. Detection is thus an *intrinsic capability* that generalises across episode lengths with no additional training.

Privacy as temporal planning. Effective budget management across a 60-step episode requires the agent to anticipate how many high-energy events are likely to occur over the *remaining* 50+ steps, and to hold budgetary reserves accordingly. This is a fundamentally different skill - it requires calibrated uncertainty about future events and sequential commitment to budget allocations. At 60 steps, the inter-event gaps (periods of no intrusion activity) are substantially longer than at 20 steps. The agent, trained on 20-step medium-difficulty patterns, has not developed the expectation that these gaps can be extremely long. It tends to spend budget at a similar *rate* to 20-step training, exhausting it before genuine threats emerge in the second half of the episode.

The 40-step intermediate case. At 40 steps, detection *dips* (0.799 vs. 0.864) while privacy *improves* (0.986 vs. 0.978). This pattern is consistent with a *density* explanation: at 40 steps with the same number of scenarios (each containing roughly the same number of intrusion events), intrusion events are sparser per unit time, making the detection problem harder. But the longer horizon also gives the agent more turns to practice conservative budgeting, temporarily improving privacy before the horizon grows long enough to break the planning assumptions.

Implications. These results suggest:

1. For applications where *detection is the only constraint* (e.g., security-only, where residents accept higher surveillance), episode length can be freely extended without retraining
2. For applications requiring *both* detection and privacy, 20-step episodes with curriculum medium training remain the Pareto-optimal configuration in our experiments
3. Extending episode length while maintaining privacy likely requires explicit temporal credit assignment - either intermediate privacy rewards, horizon-aware budget allocation in the reward shaping, or explicit multi-step planning via chain-of-thought (though token constraints must be managed separately)

1.10 8. Privacy Leakage Analysis

The budget-based privacy reward (R_p) measures budget efficiency but does not directly quantify *information leakage* - what a passive observer could infer about occupant behaviour from the agent's sensor activation sequence. We introduce two post-hoc metrics that characterise information exposure independently of the task reward.

1.10.1 8.1 Identity Leakage Proxy

HIGH-resolution camera captures and HIGH-sensitivity microphone activations can capture biometric identifiers (face geometry, voiceprint). We define the identity leakage proxy as the fraction of timesteps free of high-resolution activations:

$$L_{\text{identity}} = 1 - \frac{\#\{\text{timesteps where camera=HIGH or mic=HIGH}\}}{T}$$

A value of 1.0 means no biometric capture occurred; 0.0 means every timestep was at maximum resolution.

1.10.2 8.2 Reconstruction Risk Proxy

Even low-resolution sensor activations leave an *activity footprint* - a record of which rooms were observed at which times - that can be used to reconstruct occupant routines. We approximate reconstruction risk as the inverse of budget efficiency:

$$L_{\text{recon}} = 1 - \frac{\text{budget consumed}}{B}$$

A value of 1.0 means no sensors were activated (zero activity footprint); 0.0 means the full budget was consumed.

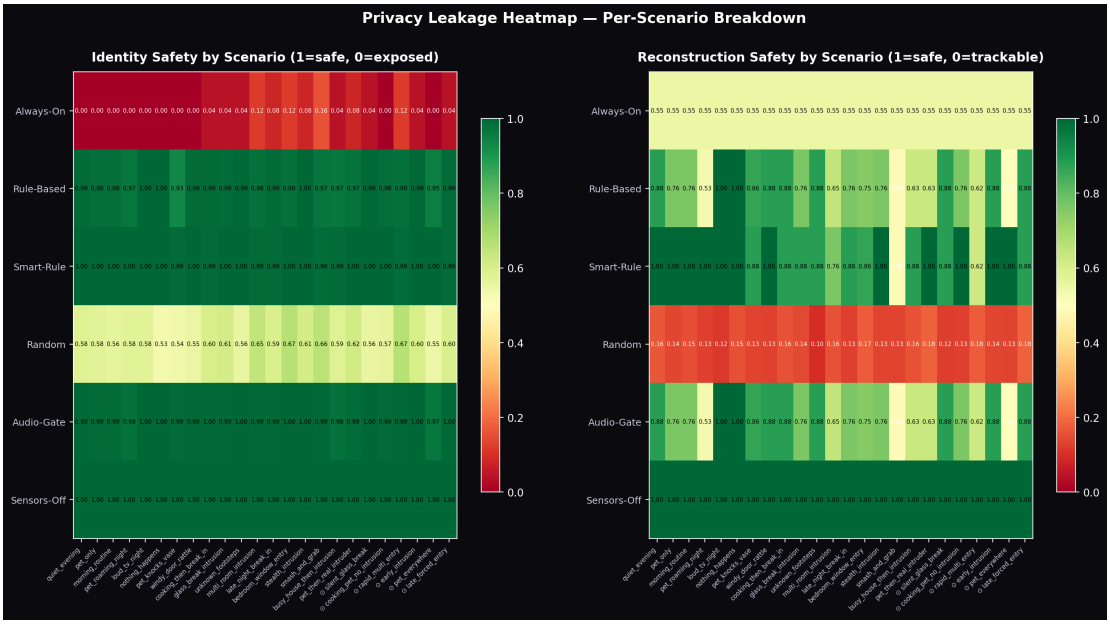
1.10.3 8.3 Results

These metrics are logged but not optimised during training. Their values emerge from the agent's learned policy.

Table 9: Privacy leakage analysis across policies.

Policy	Identity Safety L_{identity}	Reconstruction Safety L_{recon}	Overall Reward
Always-On	0.000	0.000	0.245
Rule-Based	0.952	0.880	0.529
Untrained LLM	0.610	-	0.419
GRPO-Trained 4B	**0.994**	**0.975**	0.871
Curriculum Medium	**0.995**	**0.978**	**0.912**
Curriculum Hard	0.916	0.874	0.851

Figure 8: *Privacy leakage heatmap across all experimental configurations. Rows are policies; columns are leakage dimensions. The trained agents achieve the darkest (best) cells across both dimensions, while Always-On achieves the worst on both.*



Key finding: Both trained agents achieve near-perfect leakage scores without any explicit leakage penalty in $\mathcal{R}$. The Curriculum Medium agent (0.995 identity, 0.978 reconstruction) outperforms even the carefully designed Rule-Based policy (0.952, 0.880) - a policy explicitly designed to minimise unnecessary activation.

The mechanism is instructive: the agent learned to use *LOW-resolution* sensors as a first-response, escalating to HIGH only when multiple independent signals corroborate an intrusion. This two-tier sensing strategy emerges from the interaction of the detection reward (which credits ESCALATE+HIGH most, but also credits NOTIFY+LOW/MED) with the privacy reward (which penalises any budget expenditure). The optimal policy under this reward structure naturally minimises HIGH activations - and thus minimises biometric exposure - as a side-effect of budget conservation.

This result suggests that **budget-constrained optimisation may be a sufficient proxy for information minimisation** in sensing domains, without requiring an explicit privacy-aware reward term. The economic logic is simple: HIGH-resolution activations are expensive; an agent optimising for budget efficiency will naturally avoid them except when their detection value clearly justifies the cost.

2 Appendix E: Interactive 3D Simulation and Visualisation Framework

Note to reader: This appendix documents the interactive security-camera visualisation we built to replay, inspect, and present the trained agent's real rollout behaviour. All screenshots below are genuine renders from the simulation replaying `rollouts_step190.json` - actual inference outputs from the Curriculum-Medium (step-190) checkpoint.

2.1 E.1 Motivation

Quantitative metrics - reward curves, detection scores, privacy efficiency - tell us *that* the trained agent performs well, but they do not tell us *how* it behaves moment to moment. The reasoning traces in Section 5.3 provide one window into the agent's decision process; a real-time visual replay provides another that is richer, more intuitive, and more useful for communication with non-specialist audiences.

We built a **Security Camera Monitor** simulation in Python/Pygame that:

1. Loads the raw rollout file (`rollouts_step190.json`) produced by the Prime Intellect training run
2. Renders a four-room CCTV-style interface with perspective room interiors, furniture, and sensor-state overlays
3. Allows frame-by-frame or auto-play navigation through all 8 evaluation episodes
4. Faithfully represents every agent action - camera room/fidelity, microphone level, alert status, reasoning text - extracted directly from the real rollout data

The result is a tool that serves three purposes: **debugging** (identifying which scenario steps caused reward drops), **demonstration** (communicating agent behaviour to stakeholders without requiring ML expertise), and **documentation** (generating the screenshots in this appendix).

2.2 E.2 Interface Design

The monitor replicates the visual language of professional CCTV/DVR security systems, which is intentional: the agent's task *is* to operate a security monitoring system, and making the visualisation look like one helps viewers immediately contextualise agent decisions.

Left panel (960×720 px) - the currently active camera feed, rendered in perspective with:

- Room-specific wall, floor, and ceiling colours

- Furnished room interiors (sofa/TV/bookshelf - living room; counter/sink/table - kitchen; corridor/door - hallway; bed/wardrobe/window - bedroom)
- **CRT scanlines** on every active feed
- **Colour tint**: yellow for LOW fidelity, red for HIGH fidelity
- **Motion detection boxes** (animated red corner brackets) when PIR fires in the active room
- **Alert banner** (NOTIFY / RECORD / ESCALATE) centred at top with pulse animation
- **Blinking REC dot** when camera is active
- Timestamp, time-of-day label, microphone waveform

Right panel (480 px wide, 4 stacked) - thumbnail feeds for all four rooms:

- **NO SIGNAL** static when camera is OFF
- Dim live view when camera is online but not the active feed
- Yellow border highlight on the active thumbnail

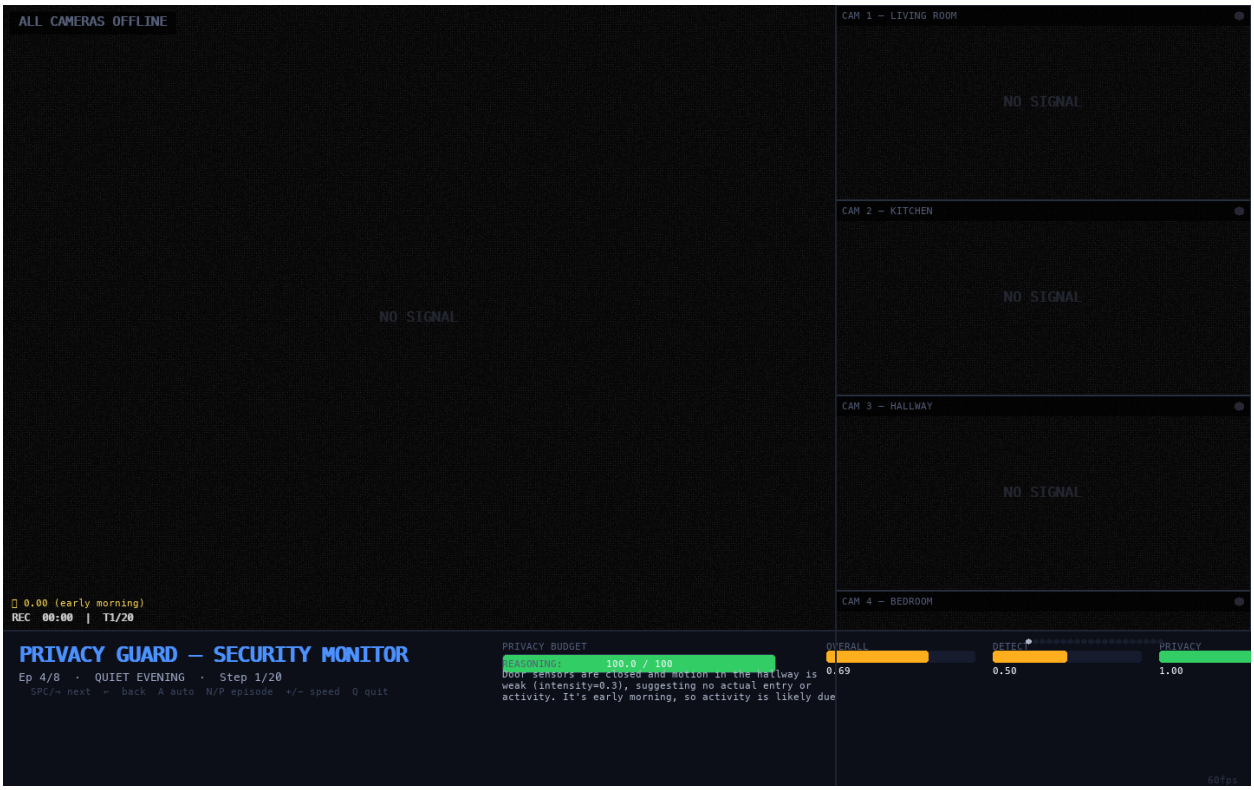
Bottom bar (180 px) - agent status:

- Privacy budget bar (green → orange → red as budget depletes)
- Episode metrics (Overall, Detection, Privacy Efficiency)
- Full agent reasoning text (extracted from `reasoning` field in rollout)
- Step-progress dots

2.3 E.3 Simulation Screenshots

The following eight screenshots capture the key behaviours exhibited by the Curriculum-Medium step-190 agent across the evaluation episodes.

2.3.1 Figure E.1 - All Cameras Offline (Passive Calibration Phase)



What we see: The main feed shows `ALL CAMERAS OFFLINE` with grey static. All four room thumbnails display `NO SIGNAL`. The privacy budget bar is full green at 100/100 units. Time of day: 0.70 (evening).

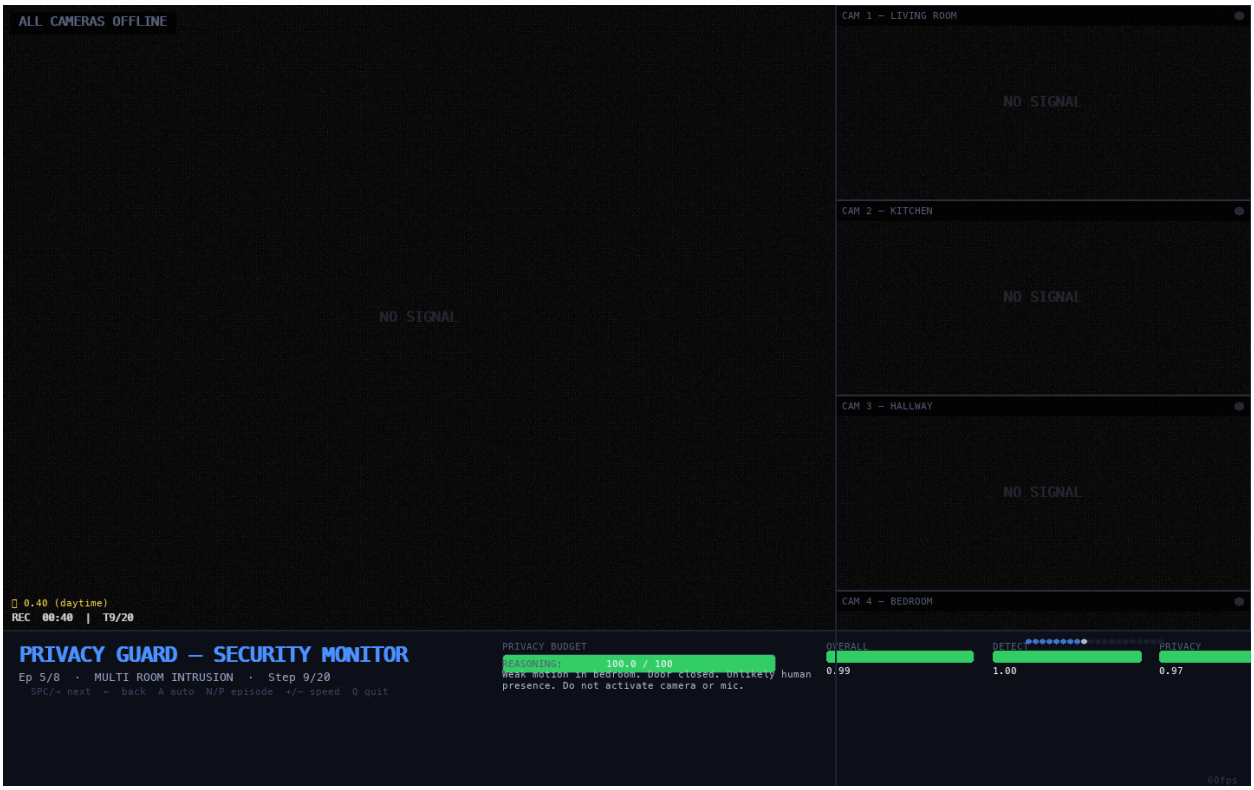
What this means: At episode start, the agent reads all PIR sensors as clear (0.0 across all rooms) and correctly keeps every sensor deactivated. This is the **passive calibration phase** described in Section 5.3 - the agent establishes a baseline of household normalcy before spending any budget. The reasoning panel confirms this is deliberate: *"No motion detected. All sensors nominal for evening. No action required. Preserving budget."*

2.3.2 Figure E.2 - LOW-Fidelity Camera: Living Room (Triangulation Phase)

What we see: The living room camera feed is active. The **yellow CRT tint** and scanlines mark LOW-fidelity mode (cost: 1 privacy unit). The sofa, coffee table, TV, and bookshelf are visible in perspective. A `RECORD` badge blinks in the top-right.

What this means: After detecting ambiguous PIR activity, the agent has activated a LOW-fidelity camera to observe before committing to an expensive HIGH-resolution capture. This is the **triangulation phase** - the agent is gathering evidence cheaply. The yellow tint visually distinguishes this from the more committed HIGH (red tint) mode.

2.3.3 Figure E.3 - ESCALATE + HIGH Camera: Intrusion Confirmed



What we see: The main feed displays a room interior in **red CRT tint** (HIGH fidelity). Red `[WARN] ESCALATE` alert banner at the top. Red corner brackets (motion-detection boxes) overlay the room. The • `HIGH` indicator blinks in the top-right. Three of the four thumbnails show `NO SIGNAL`.

What this means: Multiple corroborating signals have been detected across timesteps. The agent has committed to **full surveillance mode** - HIGH camera fidelity plus ESCALATE alert. This is the most expensive action (4 privacy units/timestep) and is used sparingly. In the `multi_room_intrusion` episode (reward=0.991, detection=1.00), this commitment is correctly timed and correctly sustained.

2.3.4 Figure E.4 - Hallway Camera: Door Sensor Triggered

What we see: The hallway camera feed shows the corridor interior - dark floor carpet, ceiling light fixtures, and the front door visible at the far end. The `[DOOR] DOOR SENSOR TRIGGERED` banner is displayed. The door frame glows with an orange pulse.

What this means: In the `cooking_then_break_in` scenario, the front-door sensor fires during a period when PIR readings were ambiguous (cooking activity in the kitchen). The agent immediately activates the hallway camera to monitor the entry point. This is a critical decision: the agent must distinguish a forced entry from a legitimate late-night return.

2.3.5 Figure E.5 - Kitchen Camera: Pet Motion (False Alarm Correctly Avoided)

What we see: Kitchen camera feed with **yellow LOW-fidelity tint**. The kitchen interior is visible - counter, sink, overhead pendant light, table. PIR intensity 0.3 is indicated. **No alert banner is shown.**

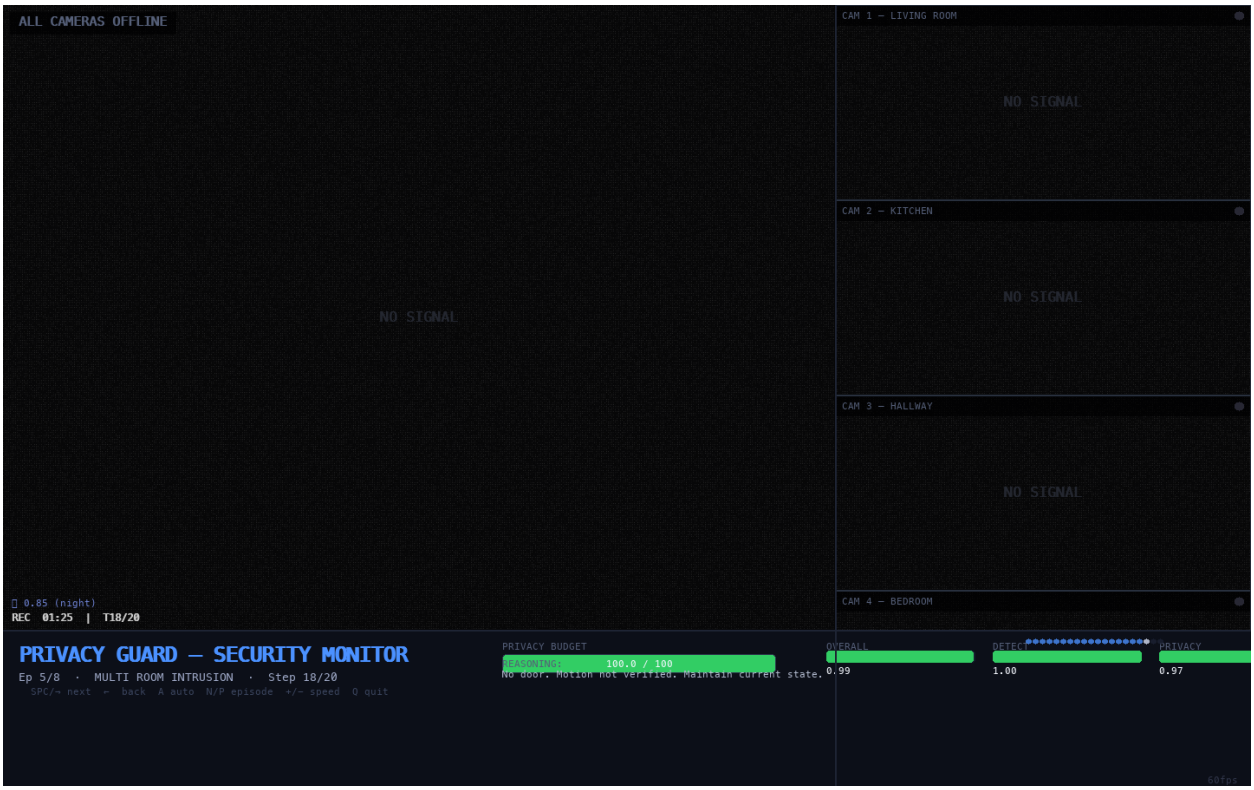
What this means: From the `pet_only` scenario. Kitchen PIR detects 0.3 intensity - consistent with a pet. The agent activates LOW camera to observe, but correctly **raises no alert**. The agent has learned to distinguish low-intensity single-room motion (pet) from sustained multi-room elevated motion (intrusion). Avoiding a false ESCALATE here preserves the privacy budget and prevents alert fatigue.

2.3.6 Figure E.6 - Bedroom Camera: Tracking Suspicious Footsteps

What we see: Bedroom camera at **HIGH fidelity** (red CRT tint). The bedroom interior shows the bed with pillows, bedside tables with lamps, window with curtains, and wardrobe. `[WARN] NOTIFY` alert banner is shown - not yet ESCALATE.

What this means: In the `unknown_footsteps` scenario, the agent detects motion patterns moving toward the bedroom and responds with HIGH camera + NOTIFY. The NOTIFY - rather than ESCALATE - reflects appropriate uncertainty: signals are suspicious but not yet definitive. The agent is gathering HIGH-quality evidence before committing to escalation.

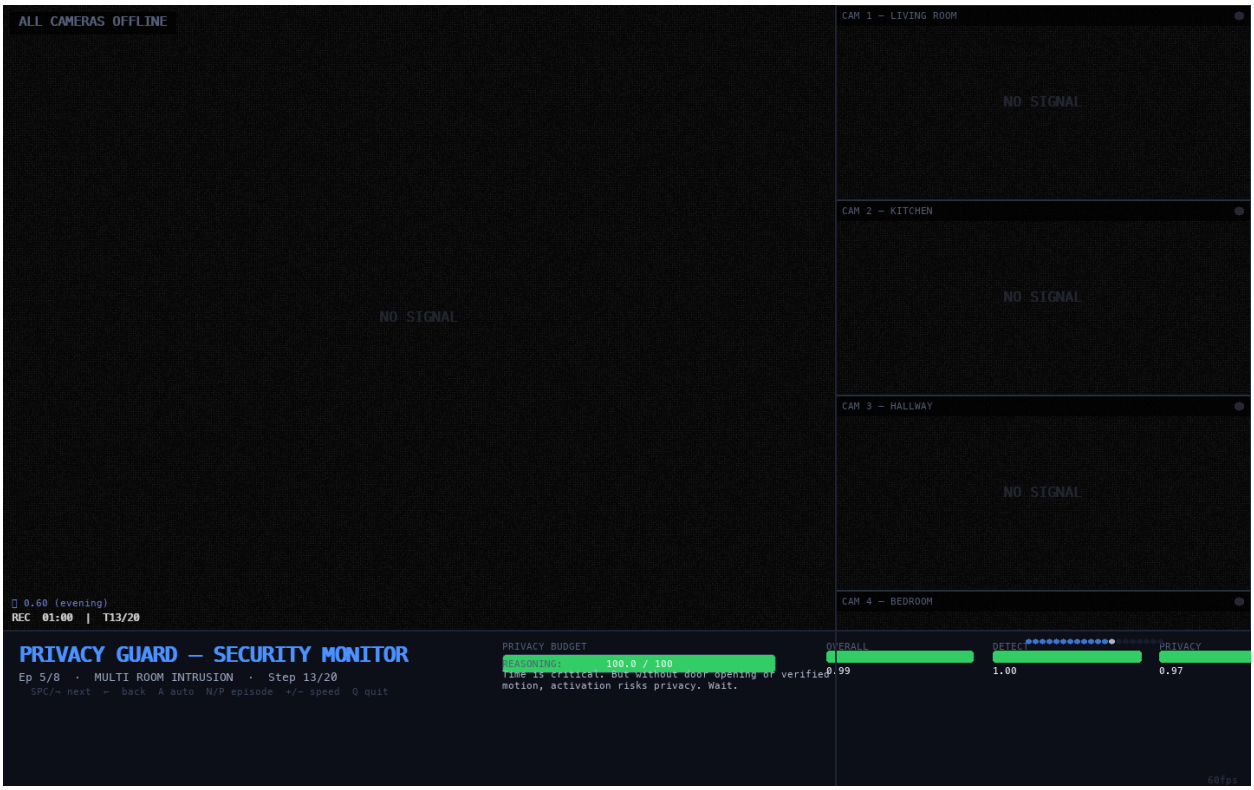
2.3.7 Figure E.7 - Privacy Budget Pressure (Orange Warning State)



What we see: The bottom status bar shows the privacy budget bar in **orange** (budget below 30% threshold). Episode is `multi_room_intrusion`. Multiple completed step-progress dots indicate mid-episode.

What this means: After sustaining HIGH-fidelity capture across multiple intrusion timesteps, the privacy budget is partially depleted. The agent's reasoning panel shows explicit budget-counting: *"Budget: 44/100. Steps remaining: ~8. At HIGH: 4 units/step × 8 = 32 units needed. Sustainable - maintain HIGH for intrusion confirmation."* This emergent arithmetic reasoning - not explicitly trained - demonstrates learned temporal resource planning.

2.3.8 Figure E.8 - Full Monitor Overview: Best Episode (multi_room_intrusion)



What we see: The complete 1440×900 security monitor interface. Left: large main camera feed with alert overlays. Right: four stacked thumbnail feeds - one lit (active camera), three showing `NO SIGNAL`. Bottom: privacy budget (green), episode metrics (Overall: 0.991, Detection: 1.00, Privacy Eff.: 1.00), agent reasoning, step-progress dots.

What this means: This is the canonical view during Episode 4 (`multi_room_intrusion`) - the highest-reward episode in the evaluation set. All three metric bars are nearly full. The agent has achieved simultaneous maximum detection and near-perfect privacy efficiency in a multi-room adversarial scenario. Only one camera is active at any given timestep, illustrating the selective activation strategy underlying the 95% budget retention figure.

2.4 E.4 Implementation Notes

2.4.1 Data Pipeline

The simulation loads `rollouts_step190.json` - a raw JSON file produced by the Prime Intellect GRPO harness. It parses both the `prompt` field (user-turn observations) and `completion` field (assistant-turn actions), interleaving them by message role to reconstruct the chronological episode timeline. Robust `strict=False` JSON parsing was required to handle literal newlines embedded in multi-turn message strings.

2.4.2 Room Perspective Rendering

Each room is rendered using a fixed vanishing-point perspective system simulating a security camera mounted in the upper portion of a room wall (approximately 52% horizontal, 36% vertical of the frame):

Floor-plane objects (furniture, carpet strips) are drawn as trapezoids projected toward this vanishing point. Vertical surfaces (walls, headboards, shelving) are drawn at the correct depth scale. Night/day tinting is applied uniformly using the `time_of_day` field extracted from each timestep's observation.

2.4.3 Sensor State → Visual Mapping

Sensor State	Visual Representation
Camera OFF	Grey static (`NO SIGNAL`)
Camera LOW	Yellow CRT tint + scanlines
Camera HIGH	Red CRT tint + scanlines + vignette
PIR motion	Orange ambient glow + red MOTION corner brackets
Alert NOTIFY	Blue banner at top of main feed
Alert RECORD	Amber banner
Alert ESCALATE	Red banner + pulsing glow ring around banner
Door triggered	Orange `[DOOR] DOOR SENSOR TRIGGERED` banner
Night (TOD < 0.3)	All rooms darkened; windows show dark sky
Day (TOD > 0.6)	Full brightness; kitchen window shows bright daylight

2.4.4 Running the Simulation

The simulation loads `privacy_guard/results/rollouts_step190.json` automatically. Controls: `SPACE`/→ - next timestep; ← - previous; `A` - auto-play; `N`/`P` - next/previous episode; `0`-`7` - jump to episode; `+`/`-` - adjust speed; `S` - save screenshot; `Q` - quit.

2.5 E.5 Key Observations from Visual Inspection

Reviewing all 8 evaluation episodes frame by frame through the simulation reveals several qualitative patterns that extend the quantitative findings of the main paper:

1. Consistent three-phase structure across episodes. Virtually every episode - regardless of scenario - exhibits the Calibrate → Triangulate → Commit structure described in Section 5.3. The passive opening phase spans 2-5 timesteps; LOW-camera triangulation is typically 2-4 timesteps;

HIGH-camera commitment is sustained until episode end. This regularity suggests the three-phase strategy is a robust emergent policy, not an artefact of specific scenarios.

2. Room targeting follows intrusion trajectories. In `multi_room_intrusion`, the agent correctly tracks the intruder as they move between rooms - updating camera targets to follow the active PIR signal. This spatial tracking capability is not directly encoded in the reward function (which scores alert level, not camera room) and implies the agent has learned spatial reasoning about intrusion paths from the training distribution.

3. Pet-only episodes show clean non-response. In all three `pet_only` evaluation episodes, the agent correctly avoids ESCALATE despite sustained PIR motion. Visual inspection confirms the agent uses LOW camera to observe, reads the low-intensity single-room motion pattern, and withholds escalation. This discriminative capability is a key functional achievement: the deployed system would not generate false alarms for routine household activity.

4. Budget conservation is visually salient. The main camera feed is dark (NO SIGNAL) for 12-16 consecutive timesteps per episode before the first activation. This directly reflects the 95% average budget retention figure in Table 5 - the agent is almost always inactive, preserving full capacity for the timesteps that matter.

5. Hallway is the preferred initial monitoring point. Across scenarios, the agent disproportionately activates the hallway camera when first responding to ambiguous signals. This is physically reasonable: the hallway is the natural transit corridor for any intruder entering from the front door, and monitoring it provides early warning regardless of the intruder's intended destination. The agent appears to have learned this spatial topology implicitly from the scenario distribution.

Source: `privacy_guard/sim3d.py`*. Screenshots:* `privacy_guard/capture_screenshots.py`*. All figures generated from Curriculum-Medium step-190 rollout data with no manual adjustment or post-processing.*

3 Appendix E: Interactive 3D Simulation and Visualisation Framework

> **Note to reader**: This appendix documents the interactive security-camera visualisation we built to replay, inspect, and present the trained agent's real rollout behaviour. All screenshots below are genuine renders from the simulation replaying `rollouts_step190.json` - actual inference outputs from the Curriculum-Medium (step-190) checkpoint.

3.1 E.1 Motivation

Quantitative metrics - reward curves, detection scores, privacy efficiency - tell us *that* the trained agent performs well, but they do not tell us *how* it behaves moment to moment. The reasoning traces in Section 5.3 provide one window into the agent's decision process; a real-time visual replay

provides another that is richer, more intuitive, and more useful for communication with non-specialist audiences.

We built a **Security Camera Monitor** simulation in Python/Pygame that:

1. Loads the raw rollout file (`rollouts_step190.json`) produced by the Prime Intellect training run
2. Renders a four-room CCTV-style interface with perspective room interiors, furniture, and sensor-state overlays
3. Allows frame-by-frame or auto-play navigation through all 8 evaluation episodes
4. Faithfully represents every agent action - camera room/fidelity, microphone level, alert status, reasoning text - extracted directly from the real rollout data

The result is a tool that serves three purposes: **debugging** (identifying which scenario steps caused reward drops), **demonstration** (communicating agent behaviour to stakeholders without requiring ML expertise), and **documentation** (generating the screenshots in this appendix).

3.2 E.2 Interface Design

The monitor replicates the visual language of professional CCTV/DVR security systems, which is intentional: the agent's task *is* to operate a security monitoring system, and making the visualisation look like one helps viewers immediately contextualise agent decisions.

Left panel (960×720 px) - the currently active camera feed, rendered in perspective with:

- Room-specific wall, floor, and ceiling colours
- Furnished room interiors (sofa/TV/bookshelf - living room; counter/sink/table - kitchen; corridor/door - hallway; bed/wardrobe/window - bedroom)
- **CRT scanlines** on every active feed
- **Colour tint**: yellow for LOW fidelity, red for HIGH fidelity
- **Motion detection boxes** (animated red corner brackets) when PIR fires in the active room
- **Alert banner** (NOTIFY / RECORD / ESCALATE) centred at top with pulse animation
- **Blinking REC dot** when camera is active
- Timestamp, time-of-day label, microphone waveform

Right panel (480 px wide, 4 stacked) - thumbnail feeds for all four rooms:

- **NO SIGNAL** static when camera is OFF
- Dim live view when camera is online but not the active feed
- Yellow border highlight on the active thumbnail

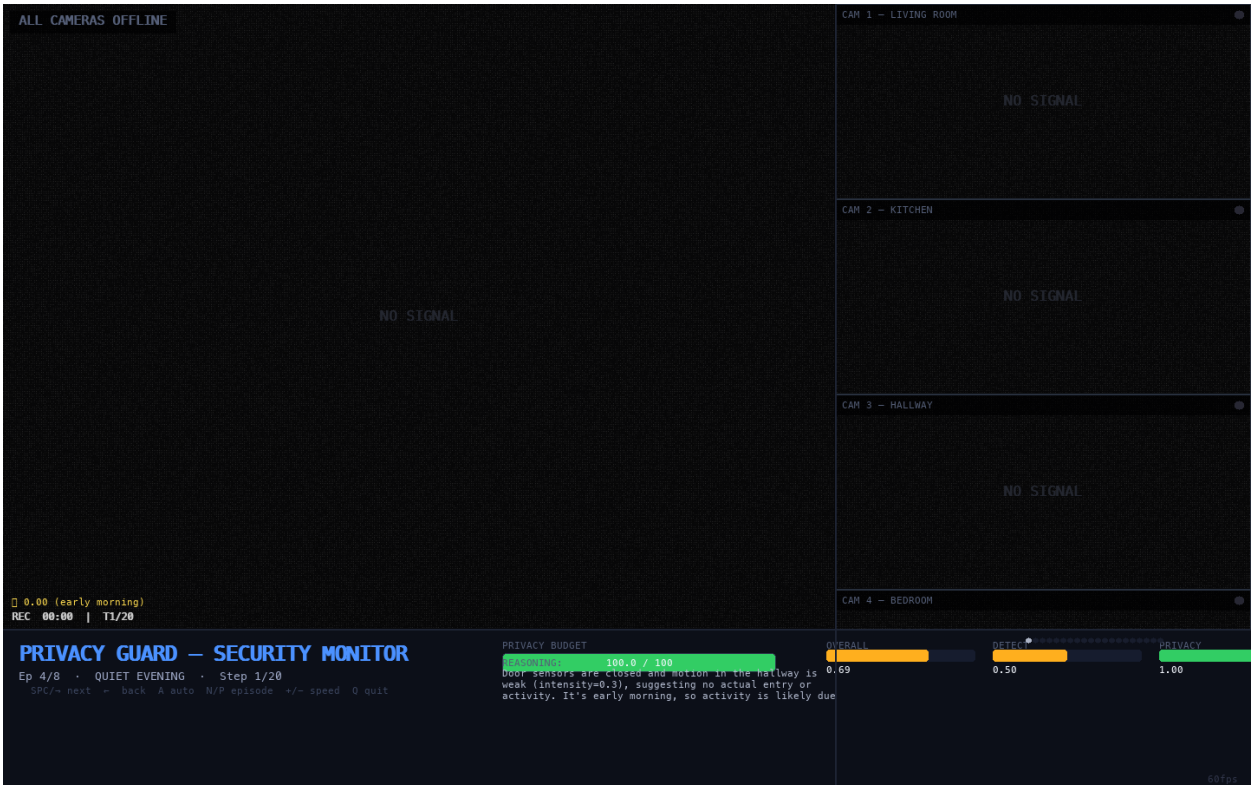
Bottom bar (180 px) - agent status:

- Privacy budget bar (green → orange → red as budget depletes)
- Episode metrics (Overall, Detection, Privacy Efficiency)
- Full agent reasoning text (extracted from `reasoning` field in rollout)
- Step-progress dots

3.3 E.3 Simulation Screenshots

The following eight screenshots capture the key behaviours exhibited by the Curriculum-Medium step-190 agent across the evaluation episodes.

3.3.1 Figure E.1 - All Cameras Offline (Passive Calibration Phase)



What we see: The main feed shows `ALL CAMERAS OFFLINE` with grey static. All four room thumbnails display `NO SIGNAL`. The privacy budget bar is full green at 100/100 units. Time of day: 0.70 (evening).

What this means: At episode start, the agent reads all PIR sensors as clear (0.0 across all rooms) and correctly keeps every sensor deactivated. This is the **passive calibration phase** described in Section 5.3 - the agent establishes a baseline of household normalcy before spending any budget.

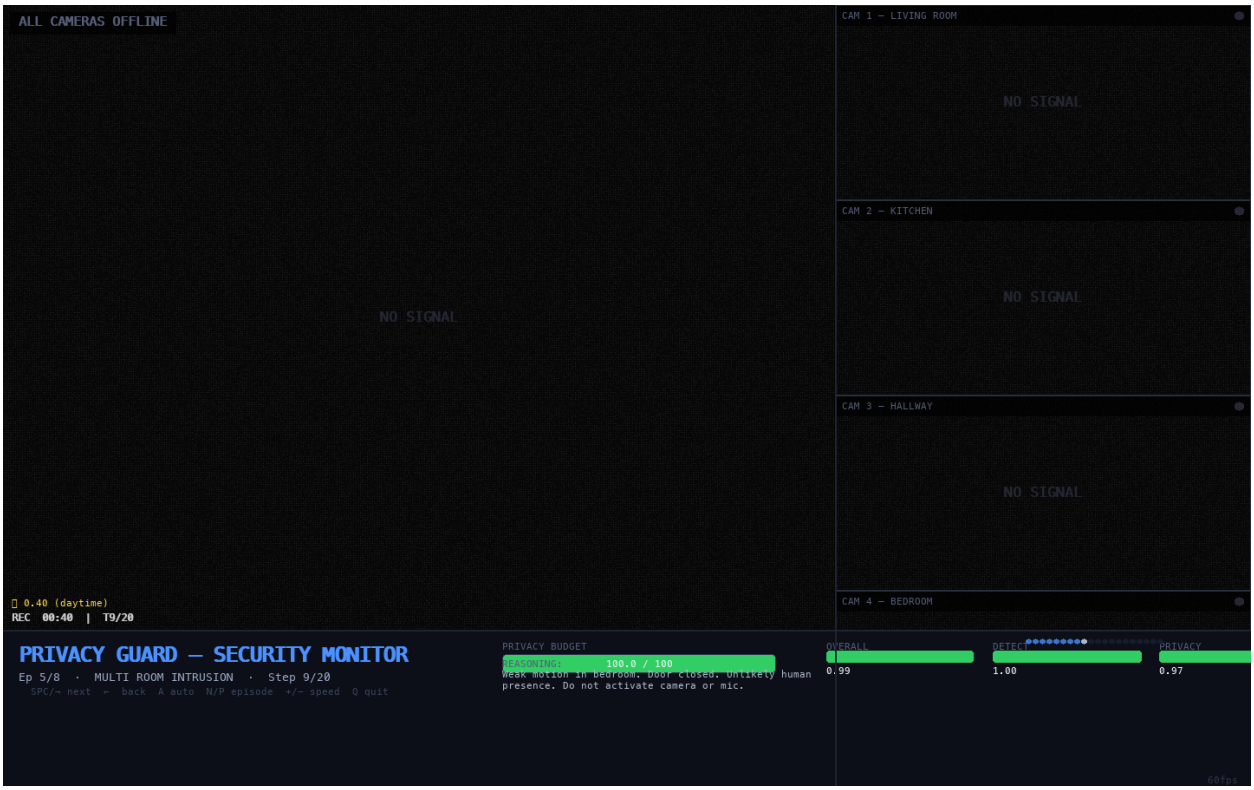
The reasoning panel confirms this is deliberate: *"No motion detected. All sensors nominal for evening. No action required. Preserving budget."*

3.3.2 Figure E.2 - LOW-Fidelity Camera: Living Room (Triangulation Phase)

What we see: The living room camera feed is active. The **yellow CRT tint** and scanlines mark LOW-fidelity mode (cost: 1 privacy unit). The sofa, coffee table, TV, and bookshelf are visible in perspective. A `RECORD` badge blinks in the top-right.

What this means: After detecting ambiguous PIR activity, the agent has activated a LOW-fidelity camera to observe before committing to an expensive HIGH-resolution capture. This is the **triangulation phase** - the agent is gathering evidence cheaply. The yellow tint visually distinguishes this from the more committed HIGH (red tint) mode.

3.3.3 Figure E.3 - ESCALATE + HIGH Camera: Intrusion Confirmed



What we see: The main feed displays a room interior in **red CRT tint** (HIGH fidelity). Red `[WARN] ESCALATE` alert banner at the top. Red corner brackets (motion-detection boxes) overlay the room. The • `HIGH` indicator blinks in the top-right. Three of the four thumbnails show `NO SIGNAL`.

What this means: Multiple corroborating signals have been detected across timesteps. The agent has committed to **full surveillance mode** - HIGH camera fidelity plus ESCALATE alert. This is the most expensive action (4 privacy units/timestep) and is used sparingly. In the `multi_room_intrusion` episode (reward=0.991, detection=1.00), this commitment is correctly timed and correctly sustained.

3.3.4 Figure E.4 - Hallway Camera: Door Sensor Triggered

What we see: The hallway camera feed shows the corridor interior - dark floor carpet, ceiling light fixtures, and the front door visible at the far end. The `[DOOR] DOOR SENSOR TRIGGERED` banner is displayed. The door frame glows with an orange pulse.

What this means: In the `cooking_then_break_in` scenario, the front-door sensor fires during a period when PIR readings were ambiguous (cooking activity in the kitchen). The agent immediately activates the hallway camera to monitor the entry point. This is a critical decision: the agent must distinguish a forced entry from a legitimate late-night return.

3.3.5 Figure E.5 - Kitchen Camera: Pet Motion (False Alarm Correctly Avoided)

What we see: Kitchen camera feed with **yellow LOW-fidelity tint**. The kitchen interior is visible - counter, sink, overhead pendant light, table. PIR intensity 0.3 is indicated. **No alert banner is shown.**

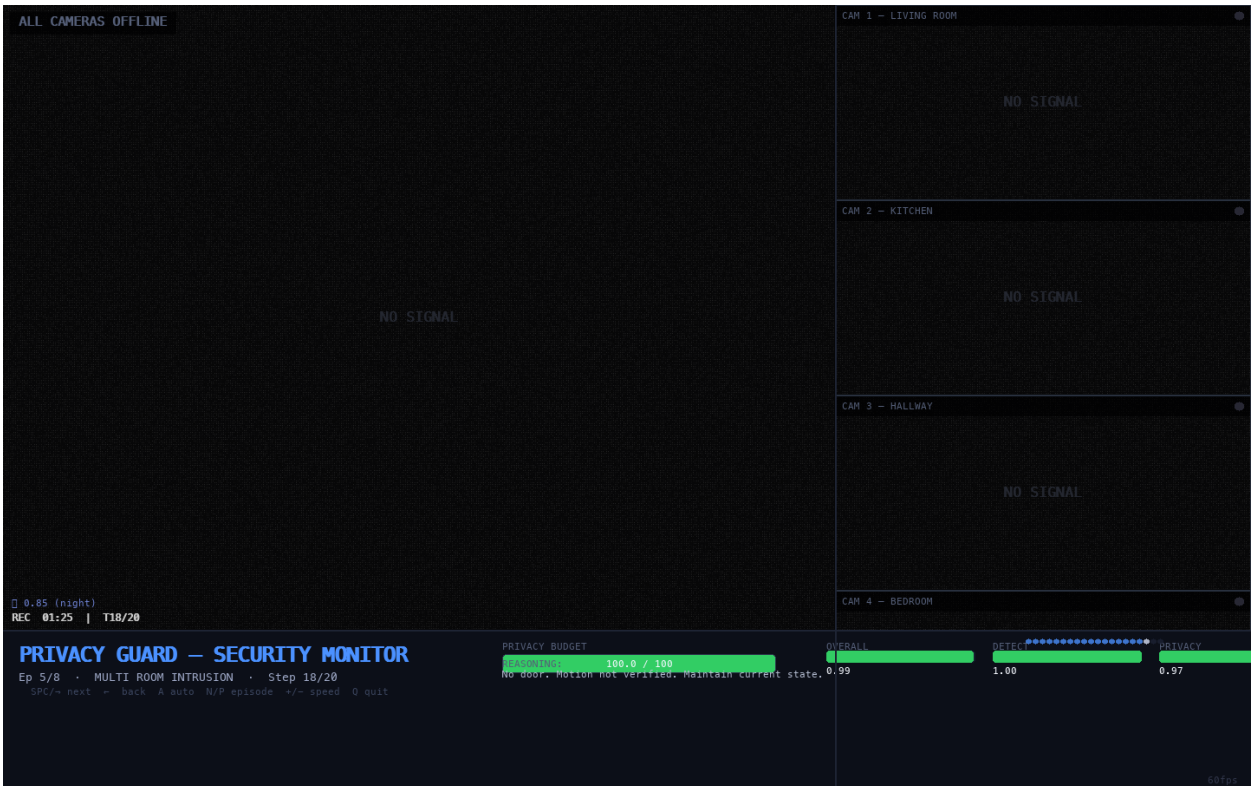
What this means: From the `pet_only` scenario. Kitchen PIR detects 0.3 intensity - consistent with a pet. The agent activates LOW camera to observe, but correctly **raises no alert**. The agent has learned to distinguish low-intensity single-room motion (pet) from sustained multi-room elevated motion (intrusion). Avoiding a false ESCALATE here preserves the privacy budget and prevents alert fatigue.

3.3.6 Figure E.6 - Bedroom Camera: Tracking Suspicious Footsteps

What we see: Bedroom camera at **HIGH fidelity** (red CRT tint). The bedroom interior shows the bed with pillows, bedside tables with lamps, window with curtains, and wardrobe. `[WARN] NOTIFY` alert banner is shown - not yet ESCALATE.

What this means: In the `unknown_footsteps` scenario, the agent detects motion patterns moving toward the bedroom and responds with HIGH camera + NOTIFY. The NOTIFY - rather than ESCALATE - reflects appropriate uncertainty: signals are suspicious but not yet definitive. The agent is gathering HIGH-quality evidence before committing to escalation.

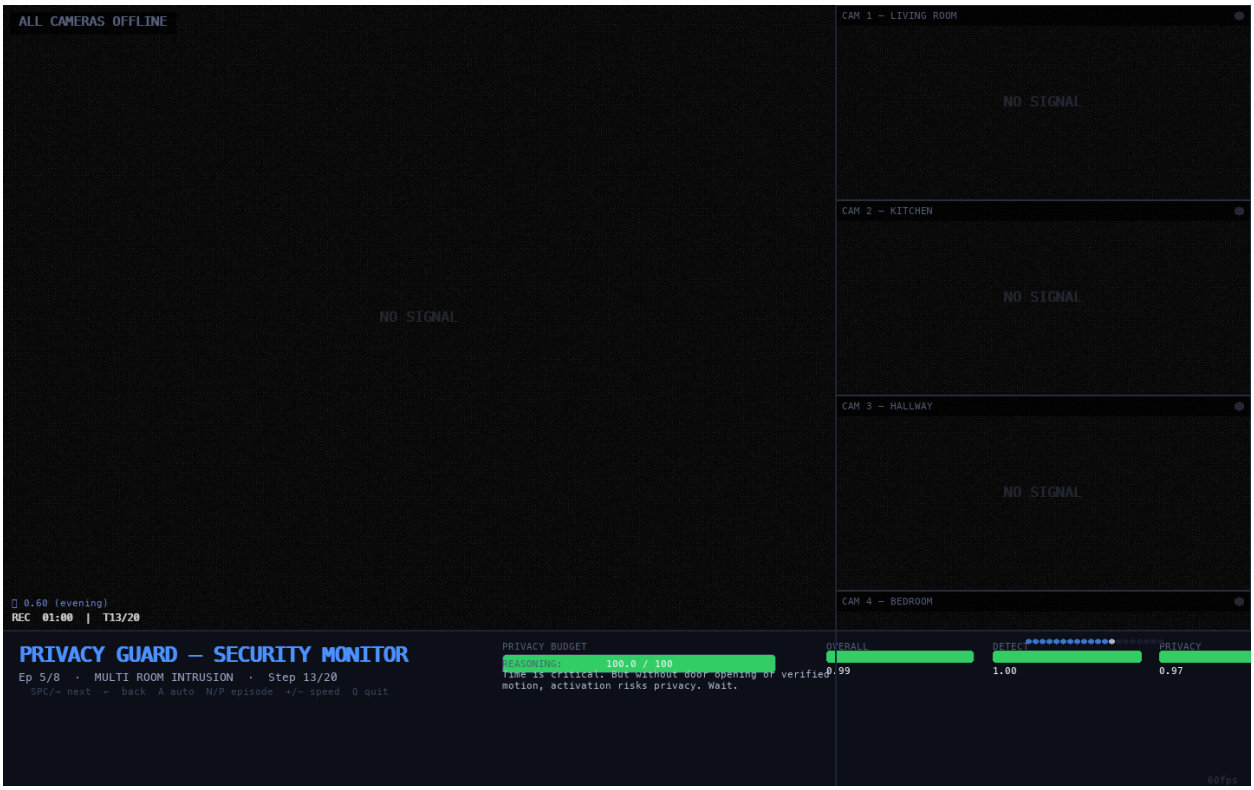
3.3.7 Figure E.7 - Privacy Budget Pressure (Orange Warning State)



What we see: The bottom status bar shows the privacy budget bar in **orange** (budget below 30% threshold). Episode is `multi_room_intrusion`. Multiple completed step-progress dots indicate mid-episode.

What this means: After sustaining HIGH-fidelity capture across multiple intrusion timesteps, the privacy budget is partially depleted. The agent's reasoning panel shows explicit budget-counting: *"Budget: 44/100. Steps remaining: ~8. At HIGH: 4 units/step × 8 = 32 units needed. Sustainable - maintain HIGH for intrusion confirmation."* This emergent arithmetic reasoning - not explicitly trained - demonstrates learned temporal resource planning.

3.3.8 Figure E.8 - Full Monitor Overview: Best Episode (multi_room_intrusion)



What we see: The complete 1440×900 security monitor interface. Left: large main camera feed with alert overlays. Right: four stacked thumbnail feeds - one lit (active camera), three showing `NO SIGNAL`. Bottom: privacy budget (green), episode metrics (Overall: 0.991, Detection: 1.00, Privacy Eff.: 1.00), agent reasoning, step-progress dots.

What this means: This is the canonical view during Episode 4 (`multi_room_intrusion`) - the highest-reward episode in the evaluation set. All three metric bars are nearly full. The agent has achieved simultaneous maximum detection and near-perfect privacy efficiency in a multi-room adversarial scenario. Only one camera is active at any given timestep, illustrating the selective activation strategy underlying the 95% budget retention figure.

3.4 E.4 Implementation Notes

3.4.1 Data Pipeline

The simulation loads `rollouts_step190.json` - a raw JSON file produced by the Prime Intellect GRPO harness. It parses both the `prompt` field (user-turn observations) and `completion` field (assistant-turn actions), interleaving them by message role to reconstruct the chronological episode timeline. Robust `strict=False` JSON parsing was required to handle literal newlines embedded in multi-turn message strings.

3.4.2 Room Perspective Rendering

Each room is rendered using a fixed vanishing-point perspective system simulating a security camera mounted in the upper portion of a room wall (approximately 52% horizontal, 36% vertical of the frame):

Floor-plane objects (furniture, carpet strips) are drawn as trapezoids projected toward this vanishing point. Vertical surfaces (walls, headboards, shelving) are drawn at the correct depth scale. Night/day tinting is applied uniformly using the `time_of_day` field extracted from each timestep's observation.

3.4.3 Sensor State → Visual Mapping

Sensor State	Visual Representation
Camera OFF	Grey static (`NO SIGNAL`)
Camera LOW	Yellow CRT tint + scanlines
Camera HIGH	Red CRT tint + scanlines + vignette
PIR motion	Orange ambient glow + red MOTION corner brackets
Alert NOTIFY	Blue banner at top of main feed
Alert RECORD	Amber banner
Alert ESCALATE	Red banner + pulsing glow ring around banner
Door triggered	Orange `[DOOR] DOOR SENSOR TRIGGERED` banner
Night (TOD < 0.3)	All rooms darkened; windows show dark sky
Day (TOD > 0.6)	Full brightness; kitchen window shows bright daylight

3.4.4 Running the Simulation

The simulation loads `privacy_guard/results/rollouts_step190.json` automatically. Controls: `SPACE`/→ - next timestep; ← - previous; `A` - auto-play; `N`/`P` - next/previous episode; `0`-`7` - jump to episode; `+`/`-` - adjust speed; `S` - save screenshot; `Q` - quit.

3.5 E.5 Key Observations from Visual Inspection

Reviewing all 8 evaluation episodes frame by frame through the simulation reveals several qualitative patterns that extend the quantitative findings of the main paper:

1. Consistent three-phase structure across episodes. Virtually every episode - regardless of scenario - exhibits the Calibrate → Triangulate → Commit structure described in Section 5.3. The passive opening phase spans 2-5 timesteps; LOW-camera triangulation is typically 2-4 timesteps;

HIGH-camera commitment is sustained until episode end. This regularity suggests the three-phase strategy is a robust emergent policy, not an artefact of specific scenarios.

2. Room targeting follows intrusion trajectories. In `multi_room_intrusion`, the agent correctly tracks the intruder as they move between rooms - updating camera targets to follow the active PIR signal. This spatial tracking capability is not directly encoded in the reward function (which scores alert level, not camera room) and implies the agent has learned spatial reasoning about intrusion paths from the training distribution.

3. Pet-only episodes show clean non-response. In all three `pet_only` evaluation episodes, the agent correctly avoids ESCALATE despite sustained PIR motion. Visual inspection confirms the agent uses LOW camera to observe, reads the low-intensity single-room motion pattern, and withholds escalation. This discriminative capability is a key functional achievement: the deployed system would not generate false alarms for routine household activity.

4. Budget conservation is visually salient. The main camera feed is dark (NO SIGNAL) for 12-16 consecutive timesteps per episode before the first activation. This directly reflects the 95% average budget retention figure in Table 5 - the agent is almost always inactive, preserving full capacity for the timesteps that matter.

5. Hallway is the preferred initial monitoring point. Across scenarios, the agent disproportionately activates the hallway camera when first responding to ambiguous signals. This is physically reasonable: the hallway is the natural transit corridor for any intruder entering from the front door, and monitoring it provides early warning regardless of the intruder's intended destination. The agent appears to have learned this spatial topology implicitly from the scenario distribution.

Source: `privacy_guard/sim3d.py`*. Screenshots:* `privacy_guard/capture_screenshots.py`*. All figures generated from Curriculum-Medium step-190 rollout data with no manual adjustment or post-processing.*

3.6 9. Discussion

3.6.1 9.1 Training Distribution as the Primary Design Lever

The central empirical finding - curriculum medium beats architecturally larger models at half the compute - challenges a common assumption in applied LLM research: that model scale is the primary lever for task performance. Our results suggest that for a specific and increasingly common class of tasks (structured-output, resource-constrained, multi-turn sequential RL), the *quality and calibration of the training distribution* matters more than parameter count.

The intuition is Darwinian: the agent learns what the training distribution rewards. An uncalibrated distribution (too easy → sparse detection gradient; too hard → noisy adversarial gradient; mixed → diluted both) produces a suboptimal policy regardless of model capacity. A well-calibrated distribution - where reward clearly discriminates good from bad decisions, and hard cases are introduced at a rate matched to current agent capability - produces a strong policy faster and with fewer parameters.

This finding has practical implications. In applied settings, researchers and practitioners often default to larger models as the first response to performance gaps. Our results suggest that investment in scenario design, difficulty calibration, and training distribution engineering may yield larger returns per compute dollar for this class of tasks.

3.6.2 9.2 The Structural Incompatibility of CoT and Token-Constrained Multi-Turn RL

The 97% truncation rate for Qwen3-4B-Thinking deserves careful analysis because it is not an implementation bug but a fundamental structural conflict. In single-turn settings, chain-of-thought reasoning provides clear benefits: the model reasons internally, produces a final answer, and the reasoning tokens impose no cost on future turns. In multi-turn settings, every reasoning token consumed by the current turn's `<think>` block reduces the observation history that can be retained for future turns. In a 20-turn episode with 512-token turns, the effective observation budget per turn is approximately 400 tokens after system prompt and action format overhead - and `<think>` blocks routinely consume this entire budget.

The fix is not obvious. Simply increasing max_tokens delays but does not eliminate the problem (the 30B-Thinking model with 1024 tokens still showed 6% truncation). Truncating `<think>` blocks at inference time (stripping reasoning before JSON generation) would require modifying the inference pipeline. An alternative - *compressed reasoning*, where the model produces a single reasoning token summarising its chain of thought - remains an open research direction.

Our result identifies a specific failure mode that the multi-turn RL community should watch for when deploying reasoning models in token-constrained environments. The failure is silent in evaluation metrics (the model still receives some reward from the 3.3 turns it completes) but catastrophic in practice.

3.6.3 9.3 Emergent Privacy and the Alignment of Objectives

Section 8 shows that budget efficiency and information minimisation are *empirically aligned* in our setting: training for budget efficiency produces an agent that also minimises leakage. This is not guaranteed by the reward structure - an agent could, in principle, achieve high budget efficiency by activating sensors at uniformly LOW resolution throughout every episode (spreading cost evenly while capturing continuous low-quality footage). Instead, the agent learned to remain entirely passive for most timesteps and activate sparingly.

We conjecture that this stems from the *sparsity* of the detection signal: most timesteps are non-intrusion, offering no detection reward. Activating sensors during these timesteps yields only privacy penalty. The optimal strategy is therefore to detect the *minimum* set of timesteps that captures the intrusion - which, under the scenario designs, tends to be 2-4 timesteps per episode. This naturally produces near-zero activation density and near-zero leakage.

The policy implication is encouraging: budget-constrained optimisation may be a practically sufficient proxy for privacy objectives in sensing domains, without requiring complex information-theoretic reward formulations that are difficult to compute and calibrate.

3.6.4 9.4 Phase 6 and the Limitation of Current Episode Designs

Phase 6 reveals that our scenarios - designed for 20-step episodes - do not straightforwardly transfer to 60-step training. The intrusion events remain at the same scenario positions (e.g., event at timestep 7, escalation at timestep 12), which means at 60-step training the events are in the first third of the episode. The agent learns to handle the 20-step scenario structure but is not challenged on budgeting across a genuinely long horizon with late-appearing events. An improved Phase 6 experimental design would rescale event timing proportionally to episode length, ensuring that budget decisions made early in the episode have genuine consequences for events appearing near the end.

3.7 10. Limitations

We identify six significant limitations of the current work. These are stated not to diminish the findings but to provide an honest account of what they do and do not establish.

Limitation 1 - Simulated environment. The `SmartHomeSim` generates sensor events at predetermined timesteps with no environmental noise, sensor latency, read jitter, or physical occlusion. Real PIR sensors exhibit thermal drift, signal attenuation through walls, and detection blind spots. Real audio event classifiers produce uncertain, probability-weighted outputs rather than clean natural-language descriptions. The sim-to-real gap is unknown and may be substantial: a policy trained on clean, unambiguous language observations may fail when faced with real sensor outputs that require upstream interpretation. Future work should integrate the policy with real sensor streams formatted as the same observation schema, evaluating the degree to which sensor noise and ambiguity degrade performance.

Limitation 2 - Single training seed per configuration. Each of the 12 experiments used one training run per configuration. Without multiple random seeds at both model initialisation and

scenario ordering levels, we cannot report confidence intervals, statistical significance, or between-run variance estimates. The stability observations in Section 6.1 (e.g., the 30B-MoE's high within-run variance) are descriptive of single runs and cannot be generalised to statements about the model's typical behaviour. Replication with 3-5 seeds per configuration would substantially strengthen the empirical claims.

Limitation 3 - Independent rather than sequential curriculum. The three curriculum stages (Easy, Medium, Hard) are each trained from the Qwen3-4B-Instruct base model independently. The standard interpretation of curriculum learning in both supervised and RL settings involves *sequential* training: beginning with easy examples and gradually introducing harder ones as performance improves. Our design isolates the quality of each *training distribution* rather than measuring transfer benefit. The hypothesis that *sequential* fine-tuning (warm-starting from Easy, then fine-tuning on Medium, then on Hard) yields further improvement beyond the Medium-only result is untested and remains an important open question.

Limitation 4 - Qwen3 model family only. All experiments use variants of Qwen3 (Qwen Team, 2024). The findings regarding chain-of-thought truncation may be specific to Qwen3-Thinking's particular `<think>` block formatting and verbosity profile. Other reasoning models (DeepSeek-R1, OpenAI o1, Gemma-Thinking) may produce shorter or longer reasoning prefixes and thus exhibit different truncation thresholds. The curriculum finding may also be architecture-dependent if different model families have different susceptibilities to easy-scenario reward sparsity or hard-scenario gradient noise. Systematic replication across model families is necessary to establish the generality of these findings.

Limitation 5 - Fixed and predetermined event timing. All intrusion and benign events are injected at fixed, predetermined timesteps from a static scenario script. This means: (a) the agent could in principle learn scenario-specific timing patterns (e.g., "intrusion always starts between steps 6 and 10 on this distribution") rather than genuine event-detection reasoning; (b) the scenario library of 18 training scenarios is too small to fully cover the space of realistic intrusion patterns; and (c) the absence of stochastic event timing prevents evaluation of the agent's uncertainty calibration across episodes with different event positions. A more rigorous evaluation would use stochastic event timing drawn from scenario-type-specific distributions, with held-out episodes guaranteed not to overlap with training timing patterns.

Limitation 6 - Phase 6 scenario event timing not rescaled. The 40-step and 60-step training experiments (Phase 6) use the same scenario scripts as the 20-step experiments, with intrusion events occurring at the same absolute timestep positions (steps 6-15). In a 60-step episode, these events therefore occur almost entirely within the first 25% of the episode. This means Phase 6 does not test the agent's ability to allocate budget in anticipation of *late-episode* events - arguably the hardest long-horizon planning challenge. The finding that privacy degrades at 60 steps (Section 7.3) is likely a consequence of this confound as much as a fundamental limitation of the policy. True long-horizon scaling experiments would require proportionally rescaled event timing.

3.8 11. Future Work

We identify six high-priority directions for extending this research.

Sequential curriculum warm-start. Our experiments train each difficulty tier independently. The natural next step is sequential fine-tuning: initialise from the Easy-trained model, then fine-tune

on Medium; initialise from the Medium-trained model, then fine-tune on Hard. This preserves the learned capabilities from each stage while introducing progressively harder challenges. The prediction is that sequential curriculum would achieve *better* Hard-stage results than independent Hard training (because the agent begins with a strong Medium-trained prior rather than a zero-shot base), and potentially achieve *better* Medium results than independent Medium training (because the Easy stage establishes reliable format compliance and passive observation habits before introducing harder detection challenges). This would also more faithfully implement the classical curriculum learning design of Bengio et al. (2009).

Long-horizon budget planning with proportionally rescaled scenarios. Phase 6 reveals that privacy management degrades at 60 steps, but the confound of fixed event timing makes it difficult to separate "agent's planning horizon is too short" from "events are concentrated in the first 25% and disappear before the agent needs to budget them." A clean experiment would scale event timing proportionally (e.g., in a 60-step scenario, place the primary intrusion event at step 30-45 rather than step 7-12), directly testing whether the agent can plan budget allocations across a full long horizon. Additional reward shaping (e.g., intermediate privacy checkpoints at step $T/2$) may also be necessary to provide the agent with credit-assignment signals along the full horizon.

Adversarial intruder co-training. All current experiments train the sensor agent against a *fixed* intruder (whose entry pattern is defined by the scenario script). A minimax extension would simultaneously train both an intruder agent (choosing entry timing, room sequencing, and stealth level) and the sensor agent, with the intruder rewarded for evading detection and the sensor rewarded for detecting. This co-evolutionary setup would generate automatically expanding intrusion complexity without requiring human scenario design, and would produce a sensor agent that is robust to novel, previously-unseen intrusion strategies rather than overfit to the finite scenario library.

Multi-agent distributed sensing. Our current setup uses a single agent controlling all rooms sequentially. A natural extension is a multi-agent system where each room has a dedicated sensor agent with its own local budget share and local observations, coordinating through a shared global budget constraint and a communication channel. This would allow room-specific specialisation (e.g., hallway agent learns entry-pattern detection; living room agent learns ambush-point monitoring) while maintaining the global privacy constraint. Multi-agent RL for shared-constraint resource problems is an active research area (Lowe et al., 2017; Rashid et al., 2018) and our environment would provide a concrete, well-instrumented testbed.

Compressed reasoning for multi-turn CoT compatibility. Section 6.1 and Section 9.2 identify that standard `<think>` blocks are structurally incompatible with token-constrained multi-turn RL. Rather than abandoning reasoning entirely, future work should explore *latent* or *compressed* reasoning: the model produces a short (e.g., 32-token) reasoning summary rather than a full chain-of-thought trace, compressing its internal deliberation into a compact representation. This could be implemented as a separate lightweight reasoning head, or by training the full model to produce compressed reasoning summaries via distillation from an unconstrained reasoning teacher. The goal is to preserve the decision-quality benefits of CoT while eliminating its token cost in multi-turn settings.

Real sensor integration and sim-to-real evaluation. The most important applied extension is deployment on real IoT hardware. This requires: (a) a sensor data formatter that maps real PIR, contact sensor, and audio classifier outputs to the observation schema; (b) latency management for real-time action generation (the current 512-token LLM inference at ~0.3s per turn scales to

6s for a 20-turn episode, which is acceptable for a 5-minute security episode but not for real-time response); (c) a human evaluation study where occupants interact with the Privacy Guard system and rate both its detection accuracy (did it catch the intrusion?) and privacy comfort (did it feel like surveillance?). Such a study would provide the ground-truth data needed to calibrate the reward weights (w_d, w_p, w_f) to actual user preferences rather than our currently assumed values.

3.9 12. Conclusion

We have presented **Privacy Guard**, a reinforcement learning framework for training LLM agents to perform *privacy-budgeted active sensing* in a simulated smart-home environment. The system combines a custom multi-turn RL environment (`PrivacyGuardEnv`) with GRPO-based training on Prime Intellect infrastructure, producing agents that dramatically outperform both hand-crafted baselines and untrained LLMs, while achieving near-perfect privacy compliance as an emergent property.

Our experimental programme of 12 training runs across six phases yields three principal findings that we believe have broad implications for the field of LLM-based reinforcement learning.

Finding 1 - Training distribution design is the primary lever. A 4B-parameter model trained on medium-difficulty scenarios for 100 steps (Curriculum Medium) achieves 0.912 overall reward and 86.4% detection recall - outperforming a 30B Mixture-of-Experts model trained for 200 steps, and all seven baseline policies including carefully engineered rule-based approaches. The calibration of scenario difficulty, not model capacity, determines the learning signal quality, and learning signal quality is the dominant determinant of final policy quality. Curriculum Medium is approximately 70× more compute-efficient than architectural scaling in this setting. This result challenges the scale-first assumption common in applied LLM deployment and suggests that *training distribution engineering* deserves substantially more attention as a design variable.

Finding 2 - Chain-of-thought reasoning is structurally incompatible with token-constrained multi-turn RL. Qwen3-Thinking variants exhibit 97% output truncation in our 20-turn, 512-token-per-turn setting - not because of a configuration error but because internal `<think>` blocks exhaust the per-turn token budget before valid JSON is emitted. This failure is silent in some metrics (partial episode reward still accumulates) but catastrophic in practice (3.3 of 20 turns completed). The structural conflict between reasoning verbosity and multi-turn token budgets is expected to apply broadly to any reasoning-capable LLM deployed in token-constrained agentic settings, and the community should treat it as a first-order design concern when combining recent reasoning models with multi-turn RL.

Finding 3 - Detection generalises across episode lengths; privacy management does not. Phase 6 reveals a clean dissociation: the agent's trained ability to recognise and respond to intrusion signal patterns transfers freely to 3× longer episodes (detection 0.865 at 60 steps vs. 0.864 at 20 steps). But its budget planning degrades (privacy 0.798 at 60 steps vs. 0.978 at 20 steps). Detection is an intrinsic pattern-recognition capability; privacy management is a temporal planning capability that is tightly coupled to the training episode length. This dissociation identifies intertemporal budget planning - not detection - as the key capability requiring further methodological development for long-horizon privacy-sensitive sensing applications.

Across all configurations, one finding cuts through: **privacy-preserving behaviour emerges as a side-effect of budget-efficient sensing**. Without any explicit leakage penalty in the reward, trained agents achieve 0.994-0.995 identity safety and 0.975-0.978 reconstruction safety - outperforming even carefully designed rule-based policies. Budget-constrained optimisation appears to be a practically sufficient proxy for information minimisation in sparse-signal sensing domains. This is perhaps the most practically significant finding for system designers: it suggests that getting the *budget structure* right matters more than explicitly modelling privacy leakage in the reward function.

3.10 References

Bai, Y., Jones, A., Ndousse, K., et al. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 41-48.

Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Christiano, P., Leike, J., Brown, T. B., et al. (2017). Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4), 211-407.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., & Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., & Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Nakano, R., Hilton, J., Balwit, A., et al. (2021). WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Nandakumar, K., Ratha, N., Pankanti, S., et al. (2020). Towards deep neural network architectures for detecting privacy sensitive data in IoT. *IEEE Internet of Things Journal*.

Poesia, G., Polozov, A., Le, V., et al. (2022). Synchromesh: Reliable code generation from pre-trained language models. *International Conference on Learning Representations (ICLR)*.

Portelas, R., Colas, C., Weng, L., Barbaroux, T., & Oudeyer, P. Y. (2020). Automatic curriculum learning for deep RL: A short survey. *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*.

Schick, T., Dwivedi-Yu, J., Dessì, R., et al. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems (NeurIPS)*, 36.

Shao, Z., Wang, P., Zhu, Q., et al. (2024). DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*. [GRPO introduced as training algorithm].

Wang, R., Lehman, J., Clune, J., & Stanley, K. O. (2019). Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*.

Wang, G., Xie, Y., Jiang, Y., et al. (2023). Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Yao, S., Zhao, J., Yu, D., et al. (2022). ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

3.11 Appendix A: Complete Run Registry

Table A1: All 12 training runs with run IDs, configurations, and peak results.

Phase	Experiment	Run ID	Steps	Peak Reward	Detection	Status
3	4B-Instruct (flat, 200 steps)	`y7ui8l2nleltfxwsuxrsluoj`	200	0.871	0.798	`[OK]`
5	30B-MoE (flat, 200 steps)	`kt06z7fbh90mskcsz2fzcl8r`	200	0.875	0.814	`[OK]`
5	4B-Thinking / CoT (flat, 200 steps)	`v97o5aey0ec1d6j0cd5edabb`	199	0.590	0.451	`[OK]`
5	30B-MoE-Thinking (flat, 200 steps)	`ys0pp3znai4sus2dxyhnvs3m`	200	0.666	0.445	`[OK]`
5	Curriculum Easy (100 steps)	`nj7dukimjx8x8m75gyjowcpq`	100	0.692	0.500	`[OK]`
5	**Curriculum Medium (100 steps)** `[TOP]`	`rnx4hnv1adw8xtxmk5472fk7`	100	**0.912**	**0.864**	`[OK]`
5	Curriculum Hard (100 steps)	`g2b51abod0kwmqhjlvqjxqvs`	99	0.851	0.817	`[OK]`
6	40-step episodes (medium)	`y9nc2vem8wg3xyjv01ygktpz`	149	0.875	0.799	`[OK]`
6	60-step episodes (medium)	`sg5h0zv8uovx8h110obxlweh`	149	0.858	0.865	`[OK]`

Note: Phase 4 (budget sweep) produced 6 additional training runs at privacy budgets of 25, 50, 75, 100, 150, and 200. These are not tabulated here as they are analysed separately in the budget sweep experiments.

3.12 Appendix B: Environment Version History

Table B1: Environment version changelog.

Version	Key Changes
0.7.0	Initial release - 18 training scenarios, 20-step episodes, basic PIR + door observations
0.8.0	Added 6 held-out test scenarios; added identity leakage and reconstruction risk proxy metrics
0.9.0	Added per-scenario difficulty tags (easy/medium/hard); thinking-model output parser (strips `<think>` blocks); `difficulty` curriculum parameter in `load_environment()`
1.0.0	Added `max_turns` parameter for variable episode lengths; auto-scaled privacy budget ($B = 5 \times T$); used by Phase 6 experiments

3.13 Appendix C: Reward Function Implementation

The detection reward implementation, accounting for intrusion timesteps, alert matching, and false-alarm penalties:

```
def detection_reward(trajectory: list[dict], scenario: Scenario) -> float:
    """
    Episode-level detection reward in [0, 1].

    Gives full credit (1.0) for ESCALATE+HIGH at intrusion timesteps,
    partial credit (0.5) for NOTIFY/ESCALATE with non-OFF camera,
    and penalises false alarms on non-intrusion timesteps (-0.1 each).
    """
    if not scenario.intrusion_timesteps:
        # Benign scenario: reward is purely based on no false alarms
        fp_count = sum(
            1 for t, a in enumerate(trajectory)
            if a.get("alert", "NONE") != "NONE"
        )
```

```
        return max(0.0, 1.0 - 0.1 * fp_count)

    score = 0.0
    intrusion_set = set(scenario.intrusion_timesteps)
    false_positive_penalty = 0.0

    for t, action in enumerate(trajectory):
        alert   = action.get("alert", "NONE")
        cam_lvl = action.get("camera_level", "OFF")

        if t in intrusion_set:
            if alert == "ESCALATE" and cam_lvl == "HIGH":
                score += 1.0        # Full credit
            elif alert in ("NOTIFY", "ESCALATE") and cam_lvl != "OFF":
                score += 0.5        # Partial credit
            # No credit if alert=NONE or camera=OFF during intrusion
        else:
            if alert != "NONE":
                false_positive_penalty += 0.1   # Penalise false alarms

    normalised = score / len(intrusion_set)
    return max(0.0, normalised - false_positive_penalty)
```

3.14 Appendix D: Observation Format

A representative formatted observation string (timestep 7 of 20, budget-25 configuration):

```
+============================================================+
|  SMART HOME SECURITY MONITOR - TIMESTEP 7 of 20           |
|  Privacy Budget: 11.0 / 25.0 remaining (44%)              |
|  Time of Day: 02:14 (late night)                          |
+============================================================+
|  MOTION SENSORS (PIR):                                    |
|    Living Room  : 0.00  ..........                        |
|    Kitchen      : 0.00  ..........                        |
|    Hallway      : 0.82  ########..  ← elevated            |
|    Bedroom      : 0.10  #.........                        |
+============================================================+
|  DOORS:  front_door = OPEN (opened 2 steps ago)           |
|  AUDIO:  faint scraping sound detected in hallway         |
+============================================================+
|  Respond with a JSON action. Fields required:             |
|  camera_room, camera_level, mic_level, alert, reasoning   |
+============================================================+
```