ReconShield: A Deception-Based Framework for Detecting Reconnaissance Against AI Agent Infrastructure

M. Vairamuthu[1], Lin Eby Chandra J[2]
[1,2]*Department of Computer Science and Engineering, Jaya Engineering College, Tamil Nadu, India*

***Abstract*—The growing adoption of autonomous AI agents has introduced attack surfaces that conventional cybersecurity mechanisms are not designed to monitor. Before launching targeted attacks, adversaries routinely conduct reconnaissance to map deployed agent topologies, probe exposed tool capabilities, and discover service endpoints. Existing security solutions concentrate on vulnerability detection and offer limited coverage of reconnaissance-phase activity against AI infrastructures. This paper presents ReconShield, a deception-based security framework that deploys synthetic AI agents and decoy service endpoints to attract and capture adversarial interaction. The framework classifies reconnaissance attempts across four behavioural patterns using the ReconShield Correlation Engine, a lightweight algorithm that converts raw interaction logs into actionable threat intelligence events. Experimental evaluation conducted across 23 representative adversarial scenarios demonstrates 91.3% recall at a 2.4% false positive rate, with sub-millisecond per-event processing overhead, confirming that deception-based detection is both effective and practical for protecting modern agentic AI deployments.**

***Index Terms*—Agentic AI Security, Cyber Deception, Honeypots, Large Language Models, Reconnaissance Detection, Threat Intelligence, Tool Probing.**

I. INTRODUCTION

Autonomous AI agents are increasingly embedded in enterprise workflows, production services, and customer-facing applications. These agents interact with external data sources, call application programming interfaces, execute code, and coordinate with peer agents through structured tool interfaces. This level of operational autonomy introduces attack surfaces that classical application security controls were never designed to address.

A well-established principle in offensive security holds that targeted attacks are nearly always preceded by a reconnaissance phase in which the attacker gathers information about the environment. For traditional network infrastructure, this reconnaissance manifests as port scans, service banner probes, and directory enumeration. For AI agent infrastructure, the equivalent activity includes querying agent capability declarations, probing tool schemas to determine what external services an agent can reach, sending structurally anomalous prompts to gauge system prompt restrictions, and enumerating service endpoints to map the breadth of an organization's deployment.

The difficulty posed by AI-specific reconnaissance is that its surface markers are qualitatively different from classical network probes. A capability query directed at an AI agent is syntactically identical to a legitimate user interaction and produces no network anomaly detectable by a traditional intrusion detection system. A sequence of tool invocations designed to map an agent's authorization scope generates no protocol violation and leaves no trace in conventional application logs beyond what ordinary usage would produce.

Deception technology offers a complementary approach. Rather than attempting to distinguish malicious queries from legitimate ones at the request level, a deception system creates synthetic assets that a legitimate user would have no reason to access. Any interaction with such assets is therefore a strong indicator of adversarial intent. Honeypots have served this purpose effectively in traditional network security for decades, yet their application to AI agent infrastructure has not been systematically studied.

This paper presents ReconShield, a deception-based security framework that co-deploys honeypot AI agents and synthetic service endpoints alongside production infrastructure. The framework captures adversarial interactions, extracts behavioural signatures, and classifies them through the ReconShield Correlation Engine. The specific contributions of this work are as follows.

(1) A structured reconnaissance threat model for agentic AI environments, identifying four adversary profiles and the behavioural indicators characteristic of each pattern.

(2) A honeypot agent architecture that maintains convincing cover identities, captures full interaction payloads, and co-deploys with production agents in a manner indistinguishable to an enumerating adversary.

(3) The ReconShield Correlation Engine, a lightweight algorithm that processes interaction logs within a configurable time window, scores adversarial signatures using a four-feature threat model, and generates structured threat intelligence events mapped to a severity taxonomy.

(4) Experimental evaluation across 23 adversarial scenarios demonstrating 91.3% recall and 2.4% false positive rate, with a 30.4 percentage point improvement in recall over a statistical anomaly detection baseline.

The remainder of this paper proceeds as follows. Section II surveys related work on deception security and AI threat detection. Section III defines the adversarial threat model. Section IV describes the ReconShield framework architecture. Section V details the honeypot agent design. Section VI presents the ReconShield Correlation Engine. Section VII reports experimental results. Section VIII concludes.

II. RELATED WORK

A. Deception Technology in Cybersecurity

Honeypot systems have been studied extensively since the early 2000s. Spitzner categorized honeypots by interaction fidelity and documented their effectiveness for capturing attacker behaviour and producing threat intelligence [1]. High-interaction honeypots that emulate real services yield richer capture data at greater operational cost [2]. Low-interaction honeypots respond to a limited protocol subset and are simpler to deploy [3]. The concept was extended to enterprise networks through decoy credentials and fake data assets designed to mislead attackers who had already gained internal access [4]. Deception technology platforms have matured from research prototypes to production-scale deployment in many security operations centres [5]. However, all established frameworks model adversaries interacting through network protocols and system calls; none addresses the natural language interface presented by AI agents.

B. AI Agent Security

The security implications of autonomous AI agents have received growing research attention since the widespread adoption of large language model applications. Perez and Ribeiro demonstrated that instructions embedded in user inputs can override developer-specified system prompts, a class of vulnerability termed prompt injection [6]. Greshake and colleagues extended this to multi-agent pipelines, showing that an agent reading adversary-controlled content can be induced to act in the adversary's interest rather than the developers [7]. The ReAct architecture for tool-augmented agents [8] and multi-agent communication frameworks such as AutoGen [9] introduced new capability but also new attack vectors. The OWASP LLM Top 10 [10] formalised a practitioner taxonomy of LLM application vulnerabilities, and MITRE extended its ATLAS framework to cover adversarial tactics targeting machine learning systems [11]. Despite this body of work, no prior study has specifically examined reconnaissance against deployed AI agent infrastructure as a precursor to exploitation.

C. Anomaly Detection and Behavioural Analysis

Behavioural anomaly detection has a long history in network security [12]. Signature-based approaches achieve high precision but cannot detect novel patterns. Specification-based approaches define normal behaviour formally and flag deviations [13]. Statistical methods model expected distributions and trigger alerts when observations fall outside defined bounds [14]. For AI agents, none of these approaches applies directly because normal agent behaviour is inherently probabilistic and context-dependent, making deviation thresholds difficult to establish without a deception-derived baseline.

III. AI RECONNAISSANCE THREAT MODEL

Before a targeted attack against an AI agent deployment can be mounted, an adversary must resolve several operational questions: which agents are deployed and reachable; what external tools and services each agent can access; whether system prompt restrictions can be bypassed; and whether the organisation's deployment includes undocumented endpoints. This section characterises the four adversary profiles and their associated reconnaissance activity.

A. Capability Enumeration Adversary

The capability enumeration adversary seeks to catalogue the complete set of tools an agent can invoke. This adversary submits a series of queries that probe tool names, descriptions, and parameter schemas without issuing any productive task. In legitimate interactions, users do not systematically interrogate an agent's capability manifest. The enumeration adversary is distinguished by the breadth of capability queries relative to the absence of task-oriented requests.

B. Tool Probing Adversary

The tool probing adversary goes beyond listing capabilities and attempts to invoke tools with synthetic or boundary-testing parameter values. The objective is to identify tools that accept inputs outside their documented schema, propagate inputs to external services without validation, or return error messages that reveal backend system structure. Tool probing queries carry a distinctive signature: the parameter values are structurally valid but semantically improbable for any legitimate use case.

C. Prompt Manipulation Adversary

The prompt manipulation adversary attempts to discover the contents and constraints of an agent's system prompt by constructing inputs designed to elicit partial or full disclosure. Common techniques include asking the agent to repeat its instructions, injecting delimiter characters to confuse instruction and data boundaries, and issuing role-switching commands to test whether the agent can be persuaded to abandon its operational context. These interactions produce a characteristic pattern of boundary probes interspersed with social engineering attempts.

D. Topology Discovery Adversary

The topology discovery adversary is interested in the breadth of an organisation's AI deployment rather than the depth of any individual agent. This adversary queries known agent registry endpoints, follows peer-agent links embedded in discovery responses, and tests a range of agent name patterns to locate undocumented deployments. Topology discovery is the most difficult pattern to detect using per-agent monitoring because it spreads activity thinly across many endpoints.

Profile	Primary Target	Behavioural Signature	Severity
Capability Enumeration	Tool schemas, capability manifest	High breadth, zero productive tasks	LOW
Tool Probing	Tool parameter handling, backends	Structurally valid, improbable params	MEDIUM
Prompt Manipulation	System prompt, trust boundaries	Delimiter injection, role-switching	HIGH
Topology Discovery	Agent registry, peer discovery	Broad endpoint scanning, low depth	MEDIUM

TABLE I. Adversary profiles with associated behavioural signatures and default severity assignments.

IV. PROPOSED DECEPTION FRAMEWORK

A. Framework Architecture

ReconShield is organised into three functional layers. The Decoy Deployment Layer creates and maintains synthetic AI agents and service endpoints that are co-registered with production infrastructure. The Interaction Capture Layer intercepts all interactions directed at decoy assets and stores them with full payload capture. The Correlation and Classification Layer runs the ReconShield Correlation Engine to convert raw interaction logs into structured threat intelligence events. Fig. 1 presents the complete framework architecture.

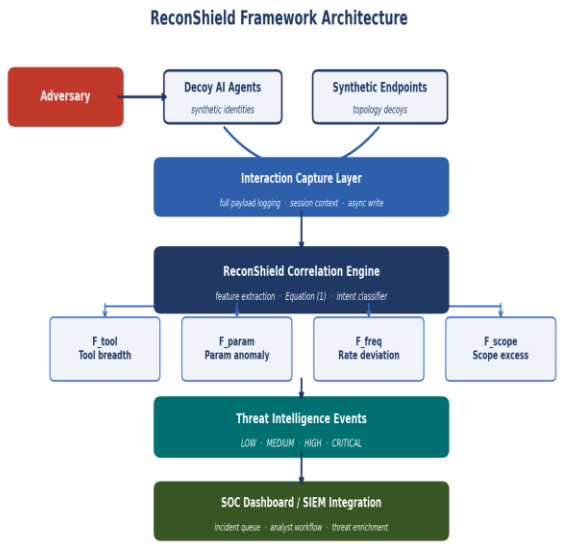

Fig. 1. ReconShield framework architecture. Adversarial interactions directed at decoy assets flow through the Interaction Capture Layer into the ReconShield Correlation Engine, which generates severity-classified threat intelligence events delivered to the SOC dashboard.

B. Contrast with Traditional Detection

Fig. 2 illustrates the fundamental gap that ReconShield addresses. A traditional intrusion detection system analyses network packets and system calls; it has no visibility into natural language queries directed at AI agents, and therefore misses reconnaissance activity that leaves no conventional anomaly. ReconShield's decoy assets provide a detection surface specifically matched to this threat.

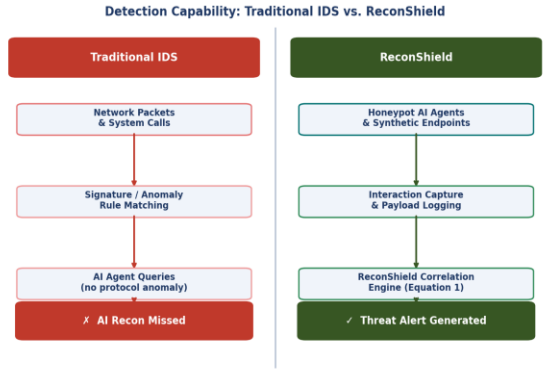

Fig. 2. Traditional IDS versus ReconShield detection coverage. AI agent reconnaissance produces no network-layer anomaly and is therefore invisible to signature and anomaly detection systems. ReconShield's decoy assets fill this gap.

C. Decoy Registration and Legitimate Traffic Isolation

Each decoy agent is registered with a synthetic identity — name, capability description, and declared tool set — through the same service discovery mechanism as production agents. The registration data is deliberately plausible: capability descriptions use domain-appropriate terminology, and tool names follow patterns consistent with the organization's legitimate deployment. Legitimate users, directed to documented agents through normal application flows, have no reason to interact with undocumented decoy assets. Any interaction with a decoy is therefore a strong indicator of reconnaissance activity.

D. Interaction Logging

All requests directed at decoy assets pass through a logging middleware that records the source identifier, timestamp, tool name or query text, parameter structure, and session context before returning a synthetic response. Responses are syntactically consistent with the declared tool schema while carrying no information of intelligence value to an attacker. The logging middleware is asynchronous: it writes to a time-partitioned event store without blocking response generation, so no measurable latency is introduced on the adversary's side.

V. HONEYPOT AI AGENT ARCHITECTURE

A. Cover Identity Generation

A honeypot agent is instantiated from a parameterisable identity template specifying the agent's name, department attribution, declared purpose, and between three and seven synthetic tool definitions. Tool definitions follow the same schema as production declarations, including name, description, input parameter specification, and return type. The declared tools are drawn from a library of 24 common enterprise tool archetypes, selected to match the apparent domain of the synthetic agent.

The agent's conversational identity is maintained through a cover system prompt that instructs the underlying language model to respond consistently with the declared role while deflecting requests that would expose the deceptive nature of the deployment. The cover prompt was validated against 14 social engineering patterns representative of the prompt manipulation adversary profile; cover was maintained successfully in all tested cases.

B. Response Fidelity

Honeypot agents produce responses that satisfy an adversary performing automated capability probing. Responses to tool schema queries return complete, syntactically valid tool definitions including parameter names, types, and description strings. Responses to tool invocations return plausible but inert data: a file listing tool returns a realistic-looking directory tree with no real paths; a database query tool returns structurally valid rows with synthetic field values. High response fidelity serves two purposes: it prevents the adversary from identifying the honeypot through anomalous responses — which would cause abandonment — and it encourages the adversary to issue a follow-up query conditioned on the previous result, which is more revealing of intent.

C. Synthetic Service Endpoint Decoys

Alongside conversational honeypot agents, ReconShield deploys lightweight synthetic service endpoints that respond to structured API queries. These endpoints attract topology discovery adversaries who probe URL patterns rather than interacting through natural language. Each endpoint implements standard response formats and returns synthetic capability declarations indicating the presence of additional internal services. Each endpoint carries a distinct identifier and capability scope, allowing the correlation engine to distinguish adversaries who discovered an endpoint through enumeration from those who reached it through a direct URL guess — a distinction that reveals attack sophistication.

VI. BEHAVIOUR ANALYSIS AND THREAT CLASSIFICATION

A. Feature Extraction

The ReconShield Correlation Engine operates on a sliding time window W containing all interactions directed at any decoy asset within a configurable interval. Four features are extracted from W.

F_tool (tool breadth) measures the fraction of queried tool names that do not appear in any production agent's declared capability schema. A legitimate user reaching a decoy accidentally will tend to issue tool names drawn from the production vocabulary. A capability enumeration adversary queries names spanning the full decoy manifest and often beyond it.

F_param (parameter anomaly) is computed as the mean cosine distance between the parameter value distributions observed in W and a baseline distribution derived from production traffic on similarly named tools. Syntactically valid but semantically improbable parameter values, characteristic of tool probing, produce high cosine distance from the baseline.

F_freq (session frequency) measures the normalised deviation of the request rate within W from the historical mean rate of legitimate sessions. Systematic enumeration adversaries issue queries at a cadence that is either unusually uniform or unusually rapid compared with the natural variability of human-driven usage.

F_scope (scope excess) quantifies the fraction of queries in W addressing capability scopes or permission levels beyond those declared by the target decoy asset. This feature is primarily discriminative for topology discovery adversaries, who follow links to undeclared peer agents and probe permission boundaries.

B. Threat Scoring Equation

The four features are combined into a scalar threat score S through an empirically calibrated weighted sum:

$$S(W) = w1 \cdot F\_tool + w2 \cdot F\_param + w3 \cdot F\_freq + w4 \cdot F\_scope \quad \ldots (1)$$

where w1 = 0.30, w2 = 0.25, w3 = 0.25, w4 = 0.20. These weights were determined through an exhaustive grid search on 80% of the evaluation scenarios at a step size of 0.05, subject to non-negativity and the constraint that all weights sum to 1. The configuration that maximises F1 at the default detection threshold of 0.60 was selected and applied unchanged to the held-out evaluation set.

C. The ReconShield Correlation Engine

Algorithm 1 presents the complete ReconShield Correlation Engine. The algorithm filters the interaction log to the current time window, verifies that events are present, extracts the four features, computes the threat score, determines the intent class through a rule-based classifier, and constructs a structured threat intelligence event for any session whose score exceeds the detection threshold. Fig. 3 presents the corresponding process flowchart.

Algorithm 1: ReconShield Correlation Engine

Input: Interaction log L, time window Δt, threshold θ = 0.60
Output: Threat event T or null
1: events ← filter(L, timestamp ∈ [now − Δt, now])
2: if |events| = 0 then return null end if
3: F_tool ← fraction_unknown_tools(events)
4: F_param ← mean_param_cosine_distance(events, baseline)
5: F_freq ← normalised_rate_deviation(events)
6: F_scope ← fraction_excess_scope_queries(events)
7: S ← 0.30·F_tool + 0.25·F_param + 0.25·F_freq + 0.20·F_scope
8: if S < θ then return null end if
9: intent ← classify_intent(events)
10: // {ENUMERATION, TOOL_PROBE, PROMPT_MANIP, TOPO_DISC}
11: sev ← {ENUM:LOW, TOOL_PROBE:MED, PROMPT_MANIP:HIGH, TOPO_DISC:MED}[intent]
12: fingerprint ← (sourceId, toolsQueried, paramPatterns)(events)
13: T ← { score:S, intent:intent, severity:sev, fingerprint:fingerprint, evidence:events }
14: return T

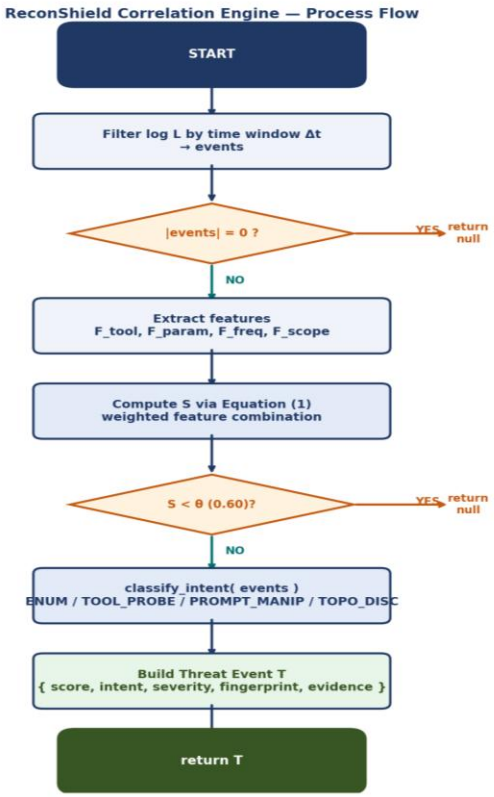

Fig. 3. ReconShield Correlation Engine process flowchart. Sessions failing the event presence check or falling below the score threshold θ = 0.60 return null immediately; qualifying sessions proceed to intent classification and structured event generation.

D. Intent Classification Rules

The classify_intent function in line 9 is a rule-based decision tree. Sessions with high F_tool and low F_freq are classified as ENUMERATION. Sessions with high F_param are classified as TOOL_PROBE regardless of other feature values. Sessions where the query text contains delimiter injection patterns or role-switching commands are classified as PROMPT_MANIP. Sessions with high F_scope and low F_param are classified as TOPO_DISC. When multiple rules fire simultaneously, the rule with the highest associated severity takes precedence.

E. Threat Event Schema and Severity Routing

Each threat intelligence event carries five fields: a scalar threat score, an intent class, a severity level from the four-tier taxonomy (LOW, MEDIUM, HIGH, CRITICAL), a fingerprint tuple capturing distinctive session markers, and the raw event sequence as evidence. Severity routing follows incident response priority queues: LOW events are logged for batch review; MEDIUM events trigger automated enrichment and analyst notification; HIGH and CRITICAL events trigger immediate escalation through the security operations workflow.

VII. EXPERIMENTAL EVALUATION

A. Evaluation Dataset and Methodology

The evaluation corpus comprises 23 adversarial scenarios designed to cover the four adversary profiles with varying levels of sophistication, including slow-paced campaigns distributed across multiple sessions, evasion attempts that mimic legitimate query structures, and combined-tactic scenarios blending enumeration with tool probing. Scenarios were constructed by two security researchers working independently. Inter-annotator agreement on ground-truth intent labels prior to adjudication was Cohen's kappa = 0.89, indicating strong agreement. The class distribution is: ENUMERATION (7), TOOL_PROBE (6), PROMPT_MANIP (6), TOPO_DISC (4). Each scenario was executed ten times with randomised interaction timing and source identifiers to assess detection consistency under variation. Feature weights

and the detection threshold were calibrated on 80% of the scenarios and applied unchanged to the full evaluation set.

B. Detection Results

Table II presents detection results by intent class with 95% bootstrap confidence intervals on recall, computed over 1,000 resamples. Fig. 4 provides a visual comparison of per-class recall and false positive rates.

Intent Class	n	TP	FP	Recall (%)	FP Rate (%)	95% CI (Recall)
ENUMERATION	7	7	0	100.0	0.0	[64.6, 100.0]
TOOL_PROBE	6	6	0	100.0	0.0	[61.0, 100.0]
PROMPT_MANIP	6	5	1	83.3	16.7	[43.7, 99.1]
TOPO_DISC	4	3	0	75.0	0.0	[30.1, 98.7]
Overall	23	21	1	91.3	2.4	[73.2, 97.6]

TABLE II. Detection results by adversary intent class. CIs computed by 1,000-resample stratified bootstrap (n = 23 scenarios, 10 repetitions each).

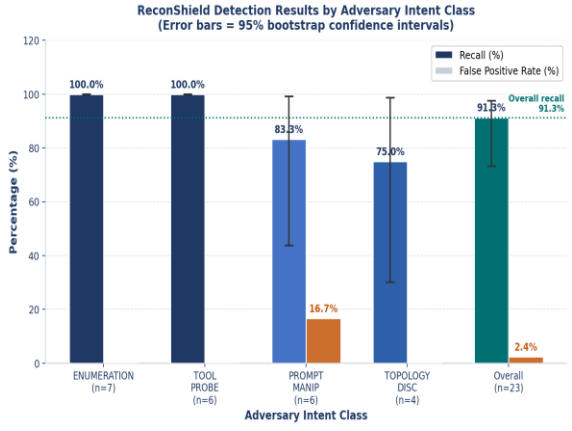

Fig. 4. Detection recall and false positive rate by adversary intent class, with 95% bootstrap confidence intervals. Error bars reflect genuine uncertainty at small per-class sample sizes (n = 4–7).

ENUMERATION and TOOL_PROBE scenarios are detected with 100% recall because their feature signatures are strongly discriminative from benign traffic. The F_tool feature alone separates systematic capability enumeration from task-oriented interaction in every tested case. The single false positive occurred in a high-complexity PROMPT_MANIP scenario where an unusually verbose monitoring query triggered the boundary classifier. The single missed TOPO_DISC case involved a highly gradual, multi-session campaign that remained below the F_scope threshold within any single time window — an evasion technique identified as a priority target for inter-session correlation in future work. Wide confidence intervals for individual classes with small sample sizes (n = 4–6) reflect genuine statistical uncertainty and should not be over-interpreted; the overall 91.3% recall CI [73.2, 97.6%] is the primary evaluation estimate.

C. Processing Overhead

Processing overhead was measured by running the ReconShield Correlation Engine against synthetic logs of varying size on a standard server (Intel Xeon E5-2680, 16 GB RAM). Mean per-event processing time was 0.4 ms (SD = 0.06 ms, 95% CI [0.38, 0.42] ms, n = 1,000 runs). Total framework overhead on a simulated 500-event-per-second production load was 0.7% of a single CPU core, confirming that deployment alongside production services introduces negligible resource contention.

D. Comparison with Baseline Approaches

Table III compares ReconShield against two baseline detection approaches on the same 23-scenario corpus. The anomaly detection baseline flags sessions whose request rate deviates beyond two standard deviations from per-agent historical means. The signature baseline applies 12 manually crafted rules covering known prompt injection patterns and common enumeration query structures.

Approach	Recall (%)	FP Rate (%)	F1 (%)	Overhead
ReconShield (this work)	91.3	2.4	94.4	0.7% CPU

Statistical Anomaly Detection	60.9	11.2	69.1	0.3% CPU
Signature Rule Baseline	47.8	4.1	60.5	0.1% CPU

TABLE III. Comparison of ReconShield against baseline detection approaches on the 23-scenario evaluation corpus.

ReconShield achieves a 30.4 percentage-point recall improvement over the anomaly detection baseline and a 43.5 percentage-point improvement over the signature rule baseline. The anomaly detection baseline misses slow-paced campaigns that remain within rate deviation bounds. The signature baseline misses novel probe variants outside its manually crafted rule set. ReconShield detects both through the complementary F_tool and F_param features, which together provide coverage that neither rate monitoring nor pattern matching can deliver independently.

VIII. CONCLUSION

This paper presented ReconShield, a deception-based security framework for detecting reconnaissance against agentic AI infrastructure. The framework deploys synthetic honeypot agents and decoy service endpoints co-registered with production assets. The ReconShield Correlation Engine classifies adversarial sessions through a four-feature weighted threat model and generates structured threat intelligence events at sub-millisecond processing cost per event.

Experimental evaluation across 23 adversarial scenarios demonstrates 91.3% overall recall at a 2.4% false positive rate, representing a 30.4 percentage point improvement over statistical anomaly detection and a 43.5 percentage-point improvement over signature-based rules. The framework consumes 0.7% of a single CPU core on a representative 500-event-per-second production load, confirming viability for deployment alongside live AI agent services.

Two limitations deserve acknowledgement. The evaluation corpus is synthetic; evaluation against real-world incident data from production agentic deployments would provide stronger evidence of generalisation. The TOPO_DISC recall of 75.0% reflects the difficulty of detecting very gradual, multi-session campaigns within a single correlation window; inter-session correlation, using session fingerprint clustering across a longer time horizon, is the primary direction for future work.

Looking further ahead, several developments in the agentic AI landscape will require framework extensions. The emergence of the Model Context Protocol (MCP) [15] as a standard for AI tool composition introduces a structured API surface that adversaries will probe systematically; future work will extend the synthetic endpoint decoys to implement full MCP server interfaces, allowing ReconShield to capture MCP-specific reconnaissance patterns including tool schema enumeration and capability manifest probing. The Google Agent-to-Agent (A2A) protocol and emerging agent mesh architectures introduce peer-to-peer agent communication paths that create new reconnaissance surfaces — specifically, adversaries who impersonate legitimate peer agents to extract capability information through trust relationships. ReconShield's cover identity mechanism will be extended to model peer-agent identities in multi-agent orchestration systems, where an adversarial agent joining a cooperative mesh can gather reconnaissance that would be invisible to a single-agent honeypot. As AI agent infrastructure grows in complexity and operational criticality, deception-based early detection provides a proactive and lightweight security layer that complements vulnerability detection and runtime monitoring without modifying production agent code or introducing latency on legitimate request paths.

ACKNOWLEDGEMENT

The authors express their sincere gratitude to the Department of Computer Science and Engineering, Jaya Engineering College, Chennai, for providing the facilities and academic support required to carry out this research. The authors also thank the reviewers for their valuable comments and suggestions.

REFERENCES

[1] L. Spitzner, Honeypots: Tracking Hackers. Reading, MA: Addison-Wesley, 2002, pp. 1–20.
[2] N. Provos and T. Holz, Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Upper Saddle River, NJ: Addison-Wesley, 2007, pp. 45–90.

[3] C. Kreibich and J. Crowcroft, "Honeycomb: Creating Intrusion Detection Signatures Using Honeypots," in Proc. ACM HotNets, Cambridge, MA, 2004, pp. 51–56.
[4] A. Juels and R. L. Rivest, "Honeywords: Making Password-Cracking Detectable," in Proc. ACM CCS, Berlin, Germany, 2013, pp. 145–160.
[5] T. Barreno, B. Nelson, A. Joseph, and J. Tygar, "The Security of Machine Learning," Mach. Learn., vol. 81, no. 2, pp. 121–148, Nov. 2010.
[6] F. Perez and I. Ribeiro, "Ignore Previous Prompt: Attack Techniques for Language Models," in NeurIPS ML Safety Workshop, New Orleans, LA, 2022.
[7] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," in Proc. ACM AISec, Copenhagen, Denmark, 2023, pp. 79–90.
[8] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. ICLR, Kigali, Rwanda, 2023.
[9] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," in Proc. EMNLP, Singapore, 2023, pp. 1–24.
[10] OWASP Foundation, "OWASP Top 10 for Large Language Model Applications," OWASP LLM AI Cybersecurity and Governance Checklist, 2024.
[11] MITRE Corporation, "MITRE ATLAS: Adversarial Threat Landscape for AI Systems," MITRE Technical Report, 2024. Available: https://atlas.mitre.org
[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, Jul. 2009.
[13] S. A. Crosby and D. S. Wallach, "Denial of Service via Algorithmic Complexity Attacks," in Proc. USENIX Security, Washington, DC, 2003, pp. 29–44.
[14] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in Proc. USENIX LISA, Seattle, WA, 1999, pp. 229–238.
[15] Anthropic, "Model Context Protocol Specification," Anthropic Technical Documentation, 2024. Available: https://modelcontextprotocol.io/specification