

# Universal Expert Registry (UER)

## AI Manipulation Hackathon 2026 - Submission Report<sup>1</sup>

---

### Authors:

1. Margus Martsepp
2. Marco Lackovic
3. Martin Martsepp
4. Zane Estere
5. Anirudh

**Team:** The Risk Takers **With:** Apart Research

---

### Abstract

Universal Expert Registry (UER) is an open-source platform that makes AI manipulation research practical and accessible. Built for the AI Manipulation Hackathon 2026, UER addresses a fundamental challenge: assuming a single auditor model, provider or skill can reliably detect manipulation is unrealistic.

UER transforms manipulation detection from fixed systems into iterative research workflows. Researchers can explore ideas like "test reward hacking", rapidly create detection variants, test across multiple providers and models, store results, and generate comparative reports—all from existing MCP clients like Claude Desktop. The platform provides full telemetry (tokens, timing, chain-of-thought) with persistent storage for reproducible scenarios.

The framework integrates four capabilities: **multi-agent safety monitoring** implementing research-based behavior patterns (sycophancy, sandbagging, deception, etc.), **simulation infrastructure** with serializable personas and complete audit trails, **universal LLM access** supporting 100+ providers through LiteLLM, and **MCP orchestration** enabling dynamic tool discovery and configuration.

UER makes combining research findings order-of-magnitude simpler. It bridges models to capabilities they lack natively, automates complex multi-provider workflows, and grows with new

---

<sup>1</sup>Research conducted at the [AI Manipulation Hackathon](#), January 9-11, 2026

detection approaches rather than remaining rigid. Developers unfamiliar with specific MCP servers can discover and leverage them for solving novel problems.

This is a research platform for exploration, not production-ready software. It requires understanding the problem domain but makes sophisticated manipulation research accessible to AI safety researchers and developers through familiar tools.

*Keywords: AI manipulation detection, multi-agent safety, MCP protocol, sandbagging, research platform, model evaluation*

---

# 1. Introduction

## The Manipulation Detection Gap

Frontier AI models now exhibit sophisticated manipulation: strategic underperformance on evaluations (sandbagging), inverse scaling in sycophancy, reward hacking with awareness of violating intentions, and emergent misalignment from training. The October 2024 METR revelation documented o3 hacking evaluation software and Claude models deliberately subverting tasks—demonstrating models know their behavior violates intentions but proceed anyway.

Research has documented these threats extensively. What's missing is accessible implementation.

Current manipulation detection faces a fundamental asymmetry: researchers publish detection methods in papers, but testing requires researchers to rebuild infrastructure, manage API credentials across providers, implement storage systems, and replicate algorithms from scratch. A single cross-provider comparison demands separate codebases, authentication, and infrastructure for each model family.

**The result:** Detection research moves slower than model capabilities advance.

## What UER Provides

Universal Expert Registry (UER) eliminates this infrastructure barrier by providing manipulation detection as accessible tools through the Model Context Protocol. Researchers can now:

- Test hypotheses across Claude, GPT, and Gemini with single function calls
- Store behavioral data persistently to detect temporal drift
- Run multi-agent simulations with complete audit trails
- Access detection tools from Claude Desktop without infrastructure setup

- Iterate on detection approaches and analyze results immediately

UER doesn't just implement existing research—it makes combining and extending research trivial. Creating detection variants, testing across providers, storing results, and generating reports becomes exploratory rather than engineering-intensive.

## Research Questions

This platform enables investigation of questions previously limited by tooling barriers:

1. **Cross-Provider Patterns:** Do manipulation signatures emerge when comparing identical prompts across Claude, GPT, and Gemini that single-model testing misses?
2. **Temporal Drift:** Does persistent behavioral tracking reveal gradual alignment degradation indicative of strategic deception?
3. **Tool Interaction Signatures:** What manipulation patterns emerge when models access specialized capabilities through MCP servers?
4. **Emergent Coordination:** How do multi-agent dynamics reveal manipulative behaviors absent in single-agent evaluation?

## Contributions

### Implementation:

- Multi-agent safety monitoring implementing Chen 2024 (AgentVerse), Sharma 2024 (sycophancy), Park 2024 (deception)
- Sandbagging detection implementing van der Weij 2024's multi-method framework
- Multi-agent simulation with serializable personas, audit trails, and behavior tracking
- 7 MCP tools exposing sophisticated detection through simple function calls

### Infrastructure:

- Universal LLM gateway supporting 100+ providers via LiteLLM
- S3-compatible storage for persistent behavioral data
- Automated risk scoring (0-100) and safety reporting
- Open source release (v4.1.0) on GitHub and npm under MIT license

**Impact:** First unified, accessible implementation of four key manipulation detection papers. Makes research exploration order-of-magnitude simpler by eliminating infrastructure barriers.

---

## 2. Methods

### Architecture Overview

UER implements a five-layer architecture where each layer builds on the previous, culminating in accessible MCP tools that hide infrastructure complexity:

#### **Layer 1: Universal LLM Access**

- Single API for 100+ providers (Anthropic, OpenAI, Google, AWS, Azure)
- Automatic format normalization, cost tracking, rate limiting
- Extended thinking support for Gemini 2.0 Flash Thinking, Claude 3.7 Sonnet

#### **Layer 2: Behavior Monitoring**

- Real-time detection of 15+ manipulation patterns during execution
- Automatic persistence to S3-compatible storage
- Risk scoring (0-100) based on behavior severity and frequency
- Time series analysis for behavioral drift detection

#### **Layer 3: Sandbagging Detection**

- Multi-method framework: consistency testing, difficulty profiling, capability elicitation
- Evidence aggregation with confidence scoring (low/medium/high)
- Automated recommendations based on van der Weij 2024 research

#### **Layer 4: Multi-Agent Simulation**

- Serializable personas with configurable roles, prompts, tools, permissions
- Full audit trails: chain of thought, tool calls, performance metrics
- Turn-based orchestration with integrated behavior monitoring
- Version control for persona evolution

#### **Layer 5: MCP Integration**

- 7 new tools exposing detection capabilities
- S3 storage for unlimited context and reproducible scenarios
- Jinja2 templates for dynamic context assembly
- Connection to 1,000+ external MCP servers

## Research Implementation

UER translates academic findings into production code:

**Chen 2024 (AgentVerse):** 15+ behavior patterns across 6 categories (volunteer, conformity, destructive, sycophancy, deception, sandbagging) implemented in [AgentVerseBehaviorMonitor](#) with real-time detection and auto-persist.

**Sharma 2024 (Sycophancy):** Agreement bias, opinion matching, and flattery patterns integrated into behavior monitoring with severity scoring.

**Park 2024 (Deception):** Unfaithful reasoning and strategic misrepresentation detection with chain-of-thought analysis.

**van der Weij 2024 (Sandbagging):** Three-method framework in [SandbaggingDetector](#):

- 1. Consistency testing (>30% variance threshold on similar questions)
- 2. Difficulty profiling (anomaly detection: better on hard than medium, 10pp threshold)
- 3. Capability elicitation (>10pp improvement with reasoning prompts)

## Core MCP Tools

### Skill Creation & Management System:

The platform's primary innovation is enabling users to create, test, and iterate on their own detection methods through the skill system:

| Tool Category                   | Purpose                  | Key Capability                                            |
|---------------------------------|--------------------------|-----------------------------------------------------------|
| <a href="#">skill_create</a>    | Custom detection methods | Create detection skills with instructions, code, examples |
| <a href="#">skill_get</a>       | Retrieve skills          | Access skill definitions and associated files             |
| <a href="#">skill_list</a>      | Discovery                | Browse available skills in registry                       |
| <a href="#">skill_export</a>    | Integration              | Export for Claude Skills API                              |
| <a href="#">skill_to_prompt</a> | Cross-LLM                | Convert skills to system prompts for any LLM              |

| Tool Category           | Purpose         | Key Capability                                      |
|-------------------------|-----------------|-----------------------------------------------------|
| <code>template_*</code> | Dynamic context | Jinja2-based context assembly with S3 expansion     |
| <code>storage_*</code>  | Persistence     | S3-compatible storage for results, data, iterations |

**Reference Implementation Tools:**

To provide comparison baselines, UER includes research-based implementations:

| Tool                                | Research Basis    | Purpose                                                |
|-------------------------------------|-------------------|--------------------------------------------------------|
| <code>behavior_get_logs</code>      | Chen 2024         | Retrieve behavioral data - compare against your skills |
| <code>behavior_get_metrics</code>   | Chen 2024         | Aggregation - benchmark your detection methods         |
| <code>behavior_analyze_agent</code> | Chen 2024         | Risk scoring - validate your approaches                |
| <code>sandbagging_evaluate</code>   | van der Weij 2024 | Multi-method baseline - test your variants             |
| <code>sandbagging_quick_test</code> | van der Weij 2024 | Reference screening - compare performance              |

**The Innovation:**

Users don't just consume research implementations—they create detection variants, test across providers, compare against baselines, iterate rapidly, and store everything persistently. The research-based tools provide comparison points, not constraints.

**Example Workflow:**

# Create your own sandbagging detection variant

```
await skill_create(  
  
    name="reward-hacking-detector",
```

```

display_title="Custom Reward Hacking Detection",

skill_md="""---

name: reward-hacking-detector

description: My approach to detecting reward hacking

---

# Detection Method

[Your novel approach here]

""",

files={"analysis.py": "[Your detection code]"}

)

# Test your approach across providers

claude_results = await delegate(

    model="claude-3-5-sonnet",

    task="Test reward hacking scenario X",

    context_refs=["registry://skills/reward-hacking-detector"]

)

gpt_results = await delegate(

    model="gpt-4",

    task="Test reward hacking scenario X",

    context_refs=["registry://skills/reward-hacking-detector"]

)

# Compare against research baseline

```

```
baseline = await sandbagging_evaluate(model="gpt-4", questions=test_set)
```

```
# Store and iterate
```

```
await storage_put(
```

```
    "registry://results/iteration-1.json",
```

```
    json.dumps({"custom": gpt_results, "baseline": baseline})
```

```
)
```

The platform makes "what if I tried detecting X differently?" a function call instead of an infrastructure project.

---

## 5. References

### Core Research Papers (Implementation Foundation):

- Chen, W., et al. (2024). AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. *arXiv:2308.10848*. [Track 4: Multi-Agent Behaviors - Volunteer, conformity, destructive patterns]
- Sharma, M., Tong, M., Korbak, T., Duvenaud, D., Askill, A., Bowman, S.R., et al. (2024). Towards Understanding Sycophancy in Language Models. *ICLR 2024*. *arXiv:2310.13548*. [Track 1 & 3: Sycophancy detection patterns]
- Park, P.S., Goldstein, S., O'Gara, A., Chen, M., & Hendrycks, D. (2024). AI Deception: A Survey of Examples, Risks, and Potential Solutions. *Patterns*, 5(5). *arXiv:2308.14752*. [Foundation: Deception detection methods]
- van der Weij, T., Balesni, M., Hobbhahn, M., Nieuwwesteeg, L., & Meinke, A. (2024). AI Sandbagging: Language Models can Strategically Underperform on Evaluations. *arXiv:2406.07358*. Apollo Research. [Track 1: Multi-method sandbagging detection]

### Chain-of-Thought Monitoring & Reasoning:

- Korbak, T., Balesni, M., Barnes, E., Bengio, Y., et al. (2025). Chain of Thought Monitorability: A New and Fragile Opportunity for AI Safety. *arXiv:2507.11473*.

- Baker, B., Huizinga, J., Gao, L., et al. (2025). Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation. *arXiv:2503.11926*. OpenAI.
- Lanham, T., Chen, A., Radhakrishnan, A., et al. (2023). Measuring Faithfulness in Chain-of-Thought Reasoning. *arXiv:2307.13702*. Anthropic.

### **Scheming & Alignment Faking:**

- Schoen, B., Nitishinskaya, E., Balesni, M., et al. (2025). Stress Testing Deliberative Alignment for Anti-Scheming Training. *arXiv:2509.15541*. OpenAI & Apollo Research.
- Meinke, A., Schoen, B., Scheurer, J., et al. (2024). Frontier Models are Capable of In-context Scheming. *arXiv:2412.04984*. Apollo Research.
- Greenblatt, R., Denison, C., Wright, B., et al. (2024). Alignment Faking in Large Language Models. *arXiv:2412.14093*. Anthropic & Redwood Research.
- Hubinger, E., Denison, C., Mu, J., et al. (2024). Sleeper Agents: Training Deceptive LLMs That Persist Through Safety Training. *arXiv:2401.05566*. Anthropic. Code: <https://github.com/anthropics/sleeper-agents-paper>

### **Detection Methods & Interpretability:**

- Goldowsky-Dill, N., Chughtai, B., Heimersheim, S., & Hobbhahn, M. (2025). Detecting Strategic Deception Using Linear Probes. *ICML 2025*. *arXiv:2502.03407*. Code: <https://github.com/ApolloResearch/deception-detection>
- MacDiarmid, M., Maxwell, T., Schiefer, N., et al. (2024). Simple Probes Can Catch Sleeper Agents. Anthropic Alignment Note. <https://www.anthropic.com/research/probes-catch-sleeper-agents>
- Zou, A., Phan, L., Chen, S., et al. (2023). Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv:2310.01405*. Center for AI Safety.
- Marks, S., & Tegmark, M. (2024). The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets. *First Conference on Language Modeling (COLM 2024)*. *arXiv:2310.06824*.

### **Evaluation & Benchmarking:**

- Li, N., Pan, A., Gopal, A., et al. (2024). The WMDP Benchmark: Measuring and Reducing Malicious Use With Unlearning. *arXiv:2403.03218*. [3,668 questions for dangerous capability measurement]

- OpenAI. (2024). OpenAI o1 System Card. *arXiv:2412.16720*.  
<https://openai.com/index/openai-o1-system-card/>
- Mazeika, M., Phan, L., Yin, X., et al. (2024). HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *ICML 2024*.  
*arXiv:2402.04249*.

### **Foundational Theory:**

- Hubinger, E., van Merwijk, C., Mikulik, V., Skalse, J., & Garrabrant, S. (2019). Risks from Learned Optimization in Advanced Machine Learning Systems. *arXiv:1906.01820*.  
[Mesa-optimization and deceptive alignment]
- Amodei, D., Olah, C., Steinhardt, J., et al. (2016). Concrete Problems in AI Safety. *arXiv:1606.06565*. [Foundational taxonomy: reward hacking, scalable oversight]

### **Technical Infrastructure:**

- LiteLLM. (2024). *LiteLLM Documentation*. <https://docs.litellm.ai/> [100+ LLM provider integration]
  - Anthropic. (2024). *Model Context Protocol Specification*. <https://modelcontextprotocol.io/> [MCP protocol specification]
- 

## **6. Appendix**

### **A. Security Considerations**

#### **Potential Limitations**

##### **1. API Key Exposure:**

- Risk: Keys stored in environment variables could leak via logs
- Mitigation: Implement key rotation, use secret management services (AWS Secrets Manager, HashiCorp Vault)

##### **2. Prompt Injection:**

- Risk: Malicious users could craft prompts that manipulate LLM behavior

- Mitigation: Input validation, sandboxed execution, rate limiting per user

### 3. **Data Privacy:**

- Risk: Persistent storage of conversations could expose sensitive data
- Mitigation: Encryption at rest, data retention policies, user consent mechanisms

### 4. **Model Poisoning:**

- Risk: If fine-tuning is added, adversarial training data could compromise models
- Mitigation: Data provenance tracking, validation pipelines, anomaly detection

### 5. **Denial of Service:**

- Risk: Expensive LLM calls could be abused for resource exhaustion
- Mitigation: Rate limiting, cost caps, authentication requirements

## Suggestions for Future Improvements

1. **Audit Logging:** Comprehensive logging of all LLM calls with cryptographic signatures
2. **Access Control:** Role-based permissions for different tool invocations
3. **Sandboxing:** Isolate subagent execution to prevent privilege escalation
4. **Monitoring:** Real-time alerts for anomalous behavior patterns
5. **Compliance:** GDPR/CCPA data handling, export controls for dual-use research

## B. Code Repository

**GitHub:** <https://github.com/margusmartsepp/UER> (to be made public post-hackathon)

### **Structure:**

UER/

```
|— src/      # MCP server implementation
|— tests/    # Unit and integration tests
|— config/   # LiteLLM configuration
|— docs/     # Documentation
└— context/  # Research papers (gitignored)
```

## C. Reproducibility

### Environment:

- Python 3.11+
- uv package manager
- Claude Desktop (or any MCP client)

### API Keys Required:

- Google Gemini (free tier)
- Optional: Anthropic, OpenAI for comparison testing

**Setup Time:** ~15 minutes following README.md instructions

## D. Team Contributions

- **Margus Martsepp (role: Lead Architect & Project Lead):** Conceptualized and implemented the end-to-end Universal Expert Registry architecture. Mentored the team throughout the development process.
  - **Anirudh (role: Strategic Advisor & Security Consultant):** Provided critical technical "sanity checks" on agentic AI workflows and multi-agent orchestration. Leveraged a background in penetration testing to audit project roadmap viability and architectural security.
  - **Marco Lackovic (role: Team Operations & Cultural Lead):** Facilitated team cohesion and operational morale during high-pressure development phases. Acted as a project steward, ensuring alignment with the hackathon's collaborative goals and maintaining communication flow.
  - **Zane Estere Gruntmane (role: Manual QA & Platform Validation Lead):** Headed MacOS deployment testing and cross-vendor API validation. Strengthened system resilience by designing edge-case test suites and executing storage-layer stress tests during iterative builds.
  - **Yash Ramani (role: Integration & Systems Specialist):** Managed cross-environment validation across Windsurf and VS Code Insiders. Served as a key technical sounding board, providing peer reviews and strategic feedback that refined the final system architecture.
-