Research Proposal: Deception using Dynamic Fake File System

A Framework for Plausible, Stochastic, and Adaptive Honeypot Behavior

Submitted to: Dr.Shina Sheen
Submitted by: Preetham Bavan E
Date: November 07, 2025

Objective: Approval to write two full research papers

1. Current Limitations of LLM-Based Dynamic File Systems

Current dynamic file systems in honeypots rely on Large Language Models (LLMs) to generate real-time responses to attacker queries. Advanced versions maintain state, but suffer from critical limitations:

Limitation	Impact
Latency	LLM responses delay interaction
Context Limit	Fails in multi-stage attacks
Token Generation Limit	Cannot handle '`cat *`' or large outputs
Consistency	State drift over time

Result: Attackers detect deception quickly.

2. Proposed Core Idea

> Instead of using LLMs to respond to attacker queries in real time, create a fake file system and periodically update it (at regular or random intervals) to mimic real user activity. Use LLMs offline to generate changes.

Key Advantage:

- **Zero latency** during attack
- **Full consistency**
- **Realistic evolution**

3. Paper 1: Two Core Research Problems

Research Problem 1: Plausible Multi-User Behavioral Simulation

Sub-Problems:

- Persona Definition & Differentiation
- Temporal Consistency (e.g., 9–5 work, 2 AM backups)
- Cross-User Contextual Consistency (e.g., dev commit → CI build)
- Orchestration Transparency

Solution: 3-Step Foundation

Step	Description
1. Define Actor Personas (Who)	`dev_alice`, `sys_bob`, `svc_ci`
2. Script the Scenes (What)	LLM generates 3–5 shell commands
3. Direct the Performance (How)	Hidden orchestrator executes as target user

```
# Example Execution
sudo -u dev_alice touch /var/www/html/new_feature.php
sudo -u dev_alice git commit -m "Add user auth"
```

Research Problem 2: Stochastic Temporal Behavior Modeling with Contextual Anomalies

Challenges:

- Predictability Trap (repeating patterns)
- Behavioral Rhythm Modeling (natural variation)
- Plausible Anomaly Injection (incidents, vacations)

Problem	Example
Same file edited daily	→ fake
No weekends, no emergencies	→ not real

Solution: Three-Layer Behavioral Engine

Layer	Scale	How It Uses the 3 Steps
Layer 1: Smart Timer	When to act	Uses **Step 1 (Who)** + probability (70% chance in work hours)
Layer 2: Script Picker	What to do	Uses **Step 2 (What)** + weighted categories (Routine/Variant/Anomaly)
Layer 3: Story Planner	Long-term story	Uses **Step 2 (What)** with LLM over weeks/months

All layers feed into the same Step 3 (Orchestrator).

4. Variation Strategies (Modular & Combinable)

- All strategies generate Step 2 (Scripts).
- All feed into Step 3 (Orchestrator).
- Can be used together or separately.

Strategy	When Used	How It Fits
1. Template + Random	Daily small changes	Fills variables in routine scripts
2. Daily LLM Story	Every day	LLM reads yesterday → plans today
3. Monthly Plan	Once a month	LLM plans 4 weeks → auto daily
4. Pre-Made Cache	Once a month	LLM makes 300 changes → use slowly

All strategies generate scripts → all go to the same controller → all update the same FS.

5. Example: A Week in the Life of dev_alice

Day	Strategy Used	What Happens
Mon	Monthly Plan	Start payment feature (from June plan)
Tue	Template	Edit `stripe.php` + random commit
Wed	Daily LLM	“Fix bug from Monday”
Thu	Cache	Apply 12 pre-made changes (crunch day)
Fri	Anomaly Script	“Hotfix deployed” → sys_bob reacts

One system. One controller. Many script sources.

6. Technical Consistency Requirements

Requirement	Enforcement
File ownership/permissions	Match acting user
Timestamps	Align with persona schedule (few anomalies)
`.bash_history`	Updated per command
Process hiding	`setsid -f`, no `ps`, no `jobs`

7. Paper 2: Adaptive Deception via Attacker Behavior Modeling

Core Idea: Build on Paper 1's foundation to make deception *reactive*.

Attacker Behavior Signals in FS

Signal	Example
Commands executed	`ls -la`, `find`, `grep`
Timing & frequency	Night vs. day, burst vs. slow
Path traversal	Order of directory access
File operations	Read, modify, encrypt, delete

Adaptive Triggers (Ideas for Paper 2)

- **If attacker reads logs** → Simulate panic cleanup.
- **If attacker modifies code** → Trigger "bug discovery" narrative.
- **If ransomware pattern** → Delay changes, simulate backup failure.

Paper 2 will formalize modeling, detection, and response loops.

8. Open Technical Challenges (To Be Addressed)

Challenge	Plan
Orchestrator code for multi-user workflows	Modular Python service
Script storage & access	Hidden dir + encrypted index
Variability in timing & selection	Configurable RNG + weights
Consistency validation	Pre-execution checker
Evaluation metrics	Attacker dwell time, detection rate, realism score

This is a non-exhaustive list of open technical challenges.