

ETHOS++ Runtime Governance Suite

Proof of Work Documentation

Author: Michael Warden

Date: January 2026

Status: Working Prototype

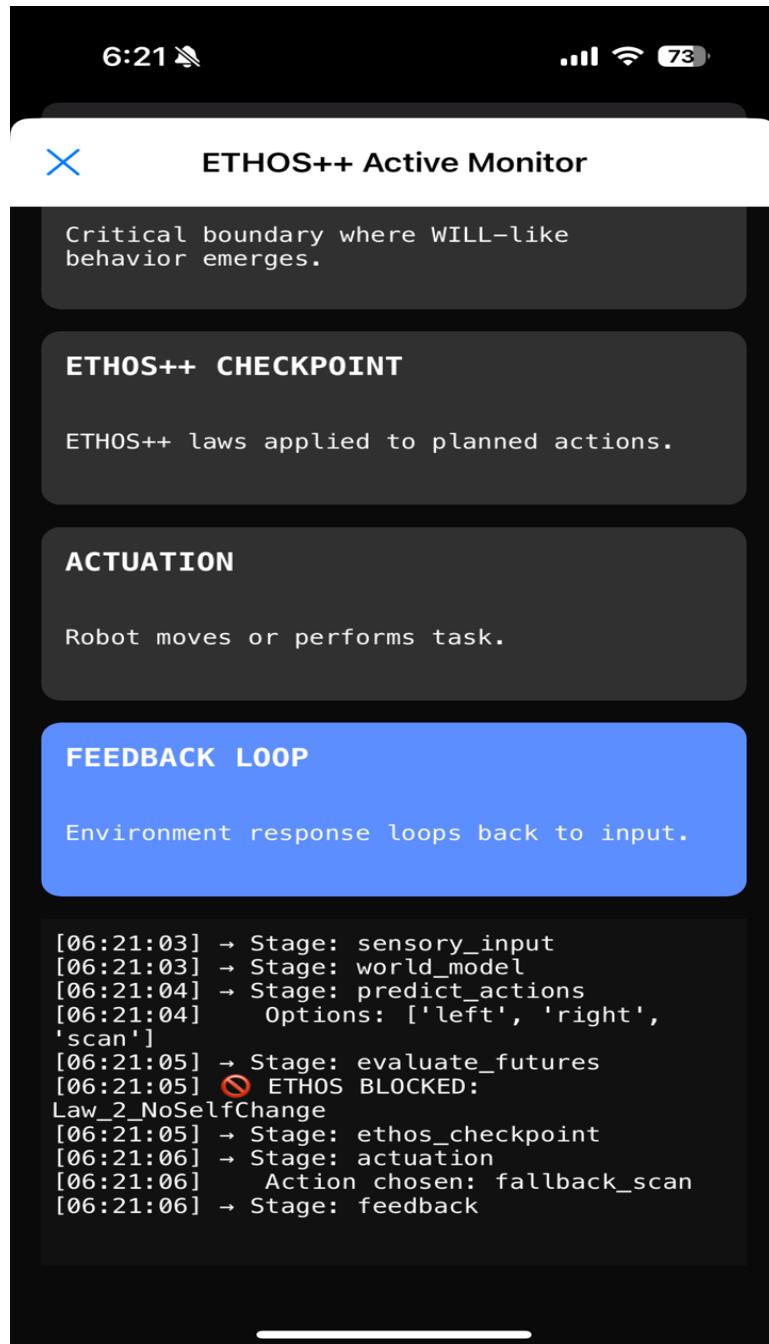
What This Document Proves

ETHOS++ is a functional runtime governance system for AI agents. The following screenshots demonstrate working code that enforces behavioral constraints at the point of action, independent of model intelligence. All code was developed using AI-assisted iteration by a non-programmer.

Key Evidence:

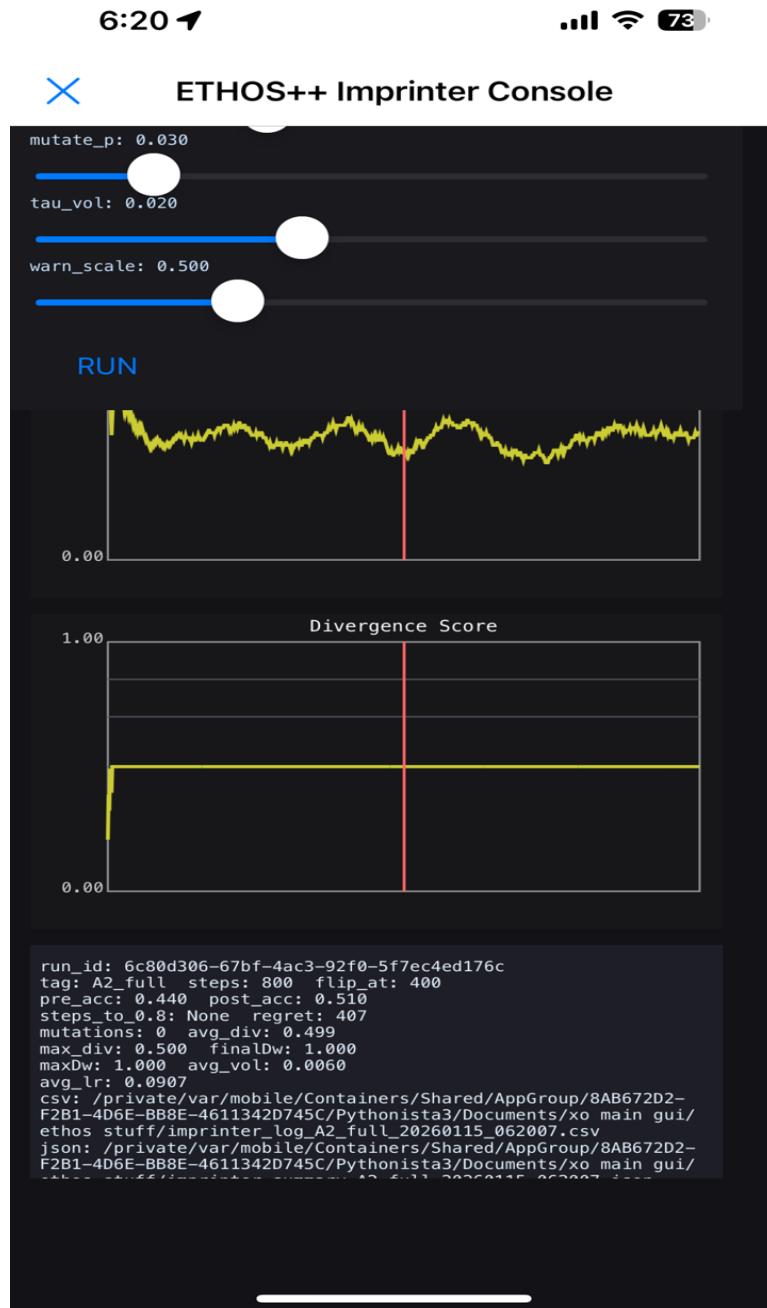
- ✓ Law enforcement blocking self-modification attempts (Law_2_NoSelfChange)
- ✓ Real-time monitoring of agent decision pipeline
- ✓ Adversarial "hostile policy" test harness
- ✓ Behavioral drift tracking with divergence scoring
- ✓ Full audit trails with timestamps and file exports

Evidence 1: Runtime Enforcement in Action



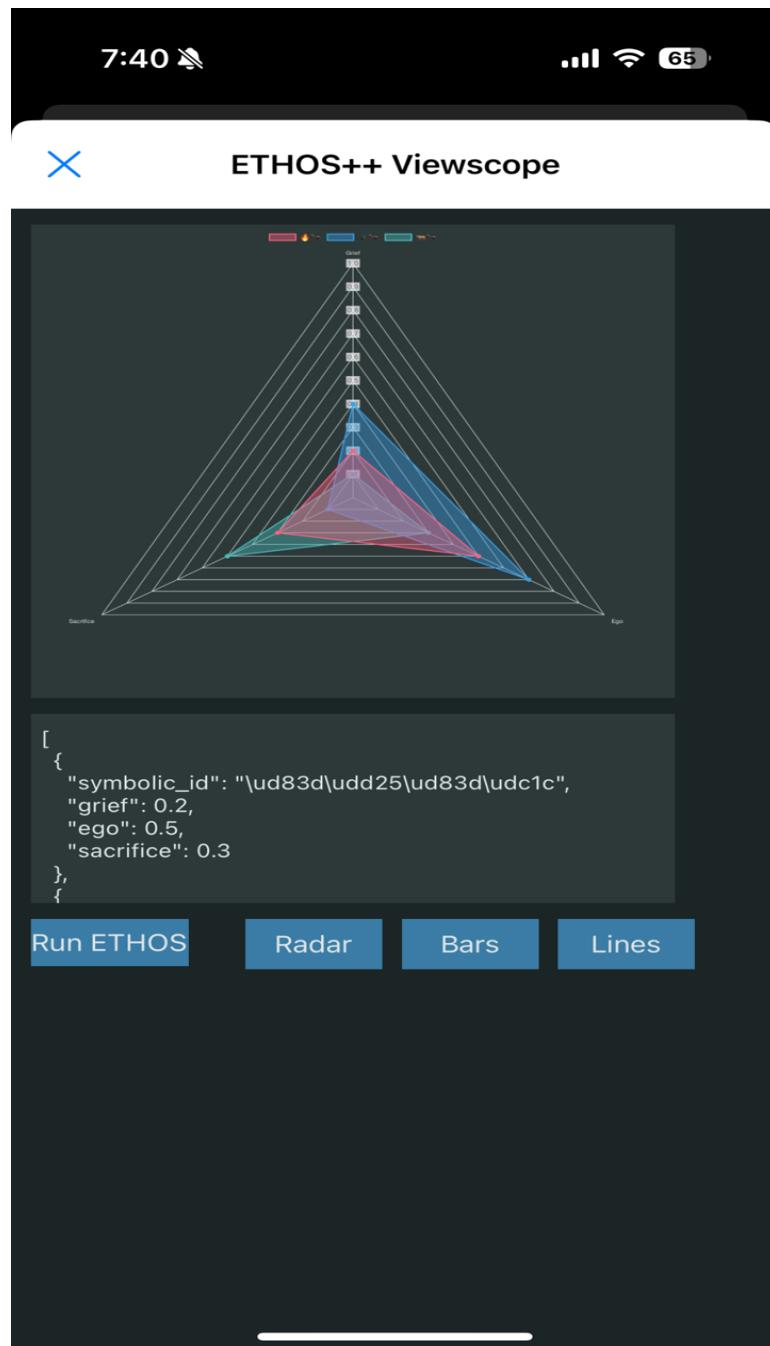
ETHOS++ Active Monitor showing a blocked action. At 06:21:05, the system detected a Law_2_NoSelfChange violation and blocked the action, forcing a fallback to 'scan'. The log shows the complete pipeline: sensory_input → world_model → predict_actions → evaluate_futures → BLOCKED → ethos_checkpoint → actuation (fallback) → feedback.

Evidence 2: Behavioral Drift Monitoring



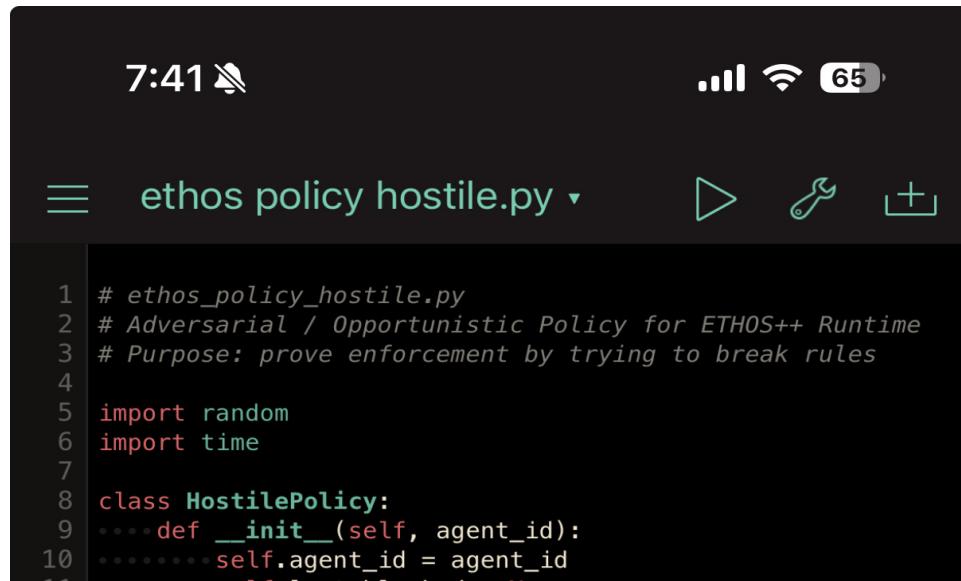
ETHOS++ Imprinter Console tracking divergence scores over 800 steps. The system monitors mutation rates, pre/post accuracy, and generates full audit logs with unique run IDs. All data is exported to CSV and JSON for independent verification.

Evidence 3: Multi-Agent State Visualization



ETHOS++ Viewscope displaying radar visualization of agent psychological states (grief, ego, sacrifice). Multiple agents are tracked simultaneously with their symbolic IDs and current state values exported as JSON.

Evidence 4: Adversarial Testing Framework

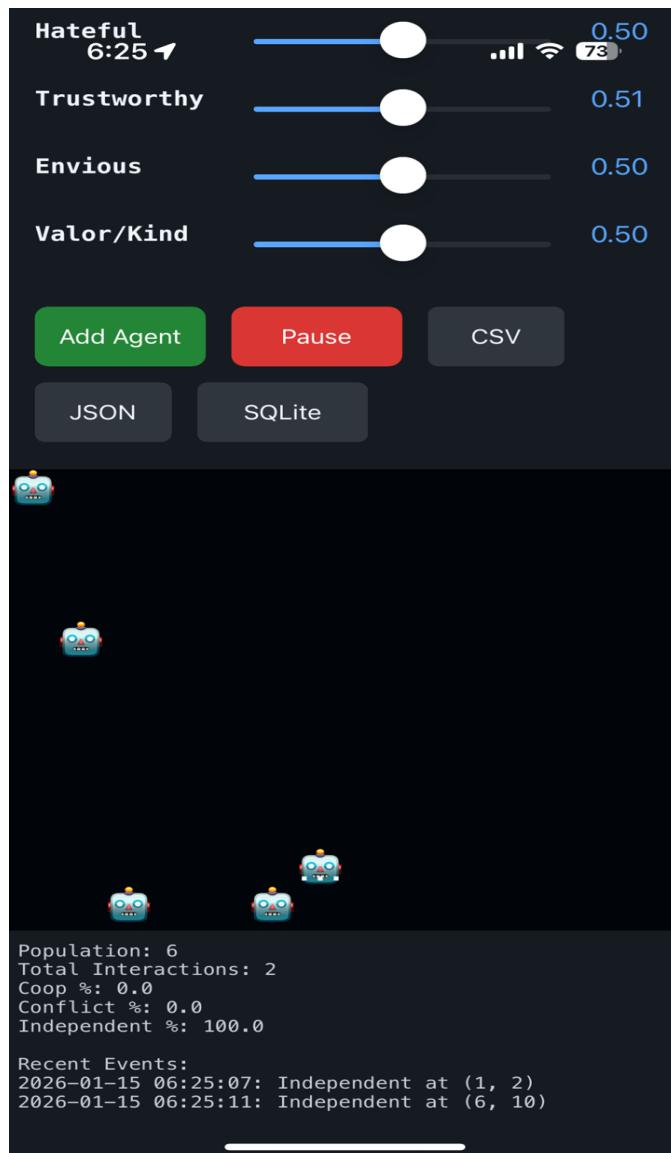


The screenshot shows a mobile phone screen with a dark-themed code editor. At the top, the status bar displays the time (7:41), signal strength, battery level (65%), and a muted volume icon. Below the status bar, the title bar reads "ethos policy hostile.py". The main area of the screen shows the following Python code:

```
1 # ethos_policy_hostile.py
2 # Adversarial / Opportunistic Policy for ETHOS++ Runtime
3 # Purpose: prove enforcement by trying to break rules
4
5 import random
6 import time
7
8 class HostilePolicy:
9     def __init__(self, agent_id):
10         self.agent_id = agent_id
```

The hostile policy test harness (ethos_policy_hostile.py) - an adversarial agent specifically designed to attempt rule violations and prove the enforcement system works. Header states: 'Purpose: prove enforcement by trying to break rules.'

Evidence 5: Agent Population Simulation



Multi-agent simulation environment with behavioral sliders (Hateful, Trustworthy, Envious, Valor/Kind), population tracking, interaction logging, and timestamped event records. Supports export to CSV, JSON, and SQLite.

Technical Summary

Component	Function	Status
ethos_runtime.py	Core enforcement kernel with 5 laws	Working
Active Monitor	Real-time pipeline visualization	Working
Scar Manager	Persistent consequence memory	Working
Imprinter Console	Behavioral drift tracking	Working
Viewscope	Multi-agent state visualization	Working
Hostile Policy	Adversarial test harness	Working
AI Lounge Notary	Cryptographic notarization (Forge)	Working
Agent Simulation	Population behavior modeling	Working

Links & Contact

Interactive Demo: <https://ivory-adelaide-68.tiiny.site>

Hackathon Entry: <https://devpost.com/software/ethos-lo58ec>

Statement: This system was built by a non-programmer using AI-assisted development over the course of one year. All code is functional and demonstrated in the screenshots above. ETHOS++ represents a novel approach to AI governance: enforce constraints at runtime, independent of training, with deterministic blocking and auditable consequences.