Closed-Loop AI-Native Threat-Driven SSDLC System

With Autonomy-Weighted Risk Scoring & Automated Threat-to-Test Transformation

A Security Operating System for Autonomous AI Systems

Closed-Loop | AI-Native | Threat-Driven | Autonomy-Scored | Self-Improving

Traditional SSDLC Cannot Secure Autonomous AI Systems

Existing security models assume deterministic software behavior

Non-Deterministic Behavior

Outputs vary per inference — traditional testing assumes repeatable results

Tool Autonomy

Agents invoke external tools with real-world side effects beyond developer control

Memory Mutation

Persistent memory enables cross-session contamination and state poisoning

Multi-Turn Reasoning

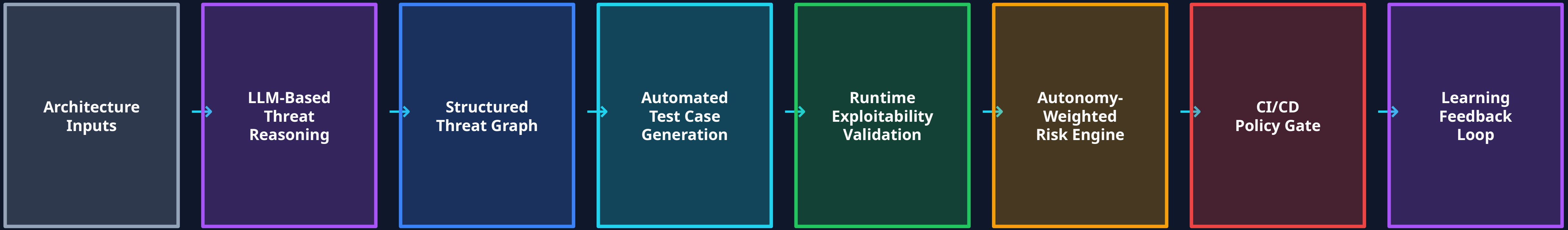
Complex reasoning chains create emergent attack surfaces invisible to static analysis

THE CORE GAP

No existing system dynamically links threat modeling, exploit validation, autonomy scoring, and CI/CD enforcement in a closed loop.

Closed-Loop Threat-Driven AI Security Engine

Core Patent System Architecture


Architecture Inputs
LLM-Based Threat Reasoning
Structured Threat Graph
Automated Test Case Generation
Runtime Exploitability Validation
Autonomy-Weighted Risk Engine
CI/CD Policy Gate
Learning Feedback Loop

CONTINUOUS FEEDBACK LOOP: Runtime signals regenerate threat models, rules, and test cases

KEY INNOVATIONS

Threat outputs automatically generate executable dynamic test scenarios

Runtime exploitability signals modify static analysis rules

Adversarial testing results auto-update agent guardrails

AI Autonomy Index dynamically weights risk computation

The novelty is in the closed-loop integration logic — not the individual components

Dynamic Risk Computation for Autonomous AI Systems

TRADITIONAL: Risk = Impact x Likelihood x Exposure   MISSING: AI Autonomy   X

Risk = Impact x Likelihood x Exposure x Autonomy Index

Autonomy Index = f( )

Tool Write Access	Irreversible Action Capability	Memory Persistence	Human-in-the-Loop Presence	Cross-Agent Interaction Scope
Can the agent modify external systems?	Can actions be rolled back?	Does the agent retain cross-session state?	Is human approval required?	Can it influence other agents?

Autonomy increases blast radius even if exploit probability is unchanged

CLAIMABLE INNOVATION — No current risk model incorporates autonomous agent capability into security scoring

Same Vulnerability, Different Risk

Vulnerability: Prompt Injection via concatenated user input | Same CVSS: 8.1

Read-Only Chatbot

Autonomy Level: LOW

No tool write access

No persistent memory

Human approves all outputs

Single-agent, no cross-system reach

RISK: 4.1 — ALLOWED

Database Write Agent

Autonomy Level: HIGH

Full database write access

Persistent memory across sessions

No human-in-the-loop

Multi-agent communication enabled

RISK: 9.2 — BLOCKED

Same vulnerability, same CVSS — fundamentally different real-world risk

The Autonomy Index captures what static severity scores cannot — the agent's capacity for irreversible, unsupervised action

Automated Threat Vector to Executable Attack Translation

Today: Threat modeling outputs documents — not executable security tests

LLM Threat Reasoning Engine Outputs

Attack Path	Full exploitation chain from entry to impact
Target Asset	Specific component, endpoint, or agent
Required Access	Authentication level and privileges needed
Exploit Pattern	Classified attack technique and payload type

→

Auto-Generated Test Outputs

Browser-Based Attack Script	End-to-end UI attack simulation
LLM Adversarial Test Case	Jailbreak and injection validation
API Fuzz Scenario	Mutation-based API exploitation
Agent Tool Abuse Test	Permission escalation validation

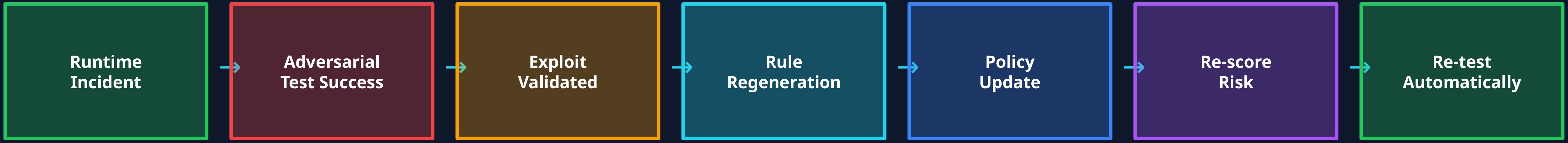
Threat Vector → Structured Graph → Test Template → Executable Attack → Result Feedback

CLAIMABLE INNOVATION

The system reasons about threats using LLM intelligence, then automatically generates executable attack simulations that validate real exploitability — a non-obvious automation beyond existing tooling.

PATENT CORE #3: CLOSED-LOOP SELF-IMPROVING SECURITY ENGINE

Closed-Loop Self-Improving Security System



← FEEDBACK LOOPS BACK — CONTINUOUS REGENERATION ←

KEY CLAIM: Runtime exploitability updates static analysis rules automatically

What Gets Regenerated Automatically:

Static Analysis Rules:	Detection patterns evolve from confirmed runtime exploits
Dynamic Test Scenarios:	Attack simulations updated with newly discovered bypass methods
Agent Guardrail Constraints:	Prompt boundaries, tool restrictions, output filters auto-updated
Autonomy Scoring Thresholds:	Risk weights recalibrated from real-world incident impact data
CI/CD Policy Gate Thresholds:	Blocking and approval thresholds adjusted from enforcement outcomes

Full automation — no manual rule updates, no human bottleneck — the loop is the invention

Patent Claims & Differentiation

CLAIM 1

A Closed-Loop AI-Native SSDLC System

A method and system comprising continuous threat modeling, automated security testing, runtime exploit validation, and feedback-driven rule regeneration operating as a unified closed-loop pipeline for securing autonomous AI systems.

CLAIM 2

An Autonomy-Weighted Dynamic Risk Engine

A computational method for dynamically scoring security risk by incorporating an Autonomy Index — a composite of tool write access, irreversible action capability, memory persistence, human-in-the-loop presence, and cross-agent interaction scope — as a multiplicative risk factor.

CLAIM 3

Threat-to-Test Automated Transformation

A method comprising: parsing architectural representations, generating structured threat vectors via LLM-based reasoning, translating vectors into executable dynamic test cases (browser attacks, adversarial prompts, API fuzz, agent abuse tests), and updating the threat graph from runtime exploitability signals.

vs. Traditional DevSecOps

Static severity scoring →

Autonomy-weighted dynamic scoring

Manual threat modeling →

LLM-generated structured threat graph

Post-deployment validation →

Continuous exploitability-driven regeneration

Filing Strategy: Narrow claims on integration logic — the novelty is the closed-loop architecture