FUZZING LABS

AI for AppSec and Offensive Security: From Automation to Autonomy

BSides Berlin 2025 - Closing Keynote - Patrick Ventuzelo

Who Am I

Patrick Ventuzelo - CEO & Founder of FuzzingLabs

Who Am I

- 10+ years in **offensive research**, **fuzzing**, and automation
- **Speaker & trainer** at Black Hat, REcon, OffensiveCon, PoC, **Pwn2Own** 2025

Who We Are

- **Deep-tech cybersecurity company** (30 + engineers) based in Paris
- Specialized in **fuzzing**, reversing, code audit & offensive AI
- Recognized **research & training** delivered worldwide

What We're Building

- FuzzForge - **AI-Native Platform** for Autonomous Vulnerability Research
- Orchestrates **multi-agent workflows** for fuzzing, reversing, and triaging
- Open-source **core + SaaS** platform for **collaborative** offensive R&D

From Automation to Autonomy

Security Automation is Evolving into Autonomy

Manual

Human-driven testing and reviews

Pentesting
Code review
Manual triage

Automated

Scripts and pipelines for repetitive tasks

CI/CD
SAST
DAST
SCA
Regression testing

AI-Driven

AI assists audits, fuzzing & triaging

LLM-assisted SAST
Harness generation
Triage automation

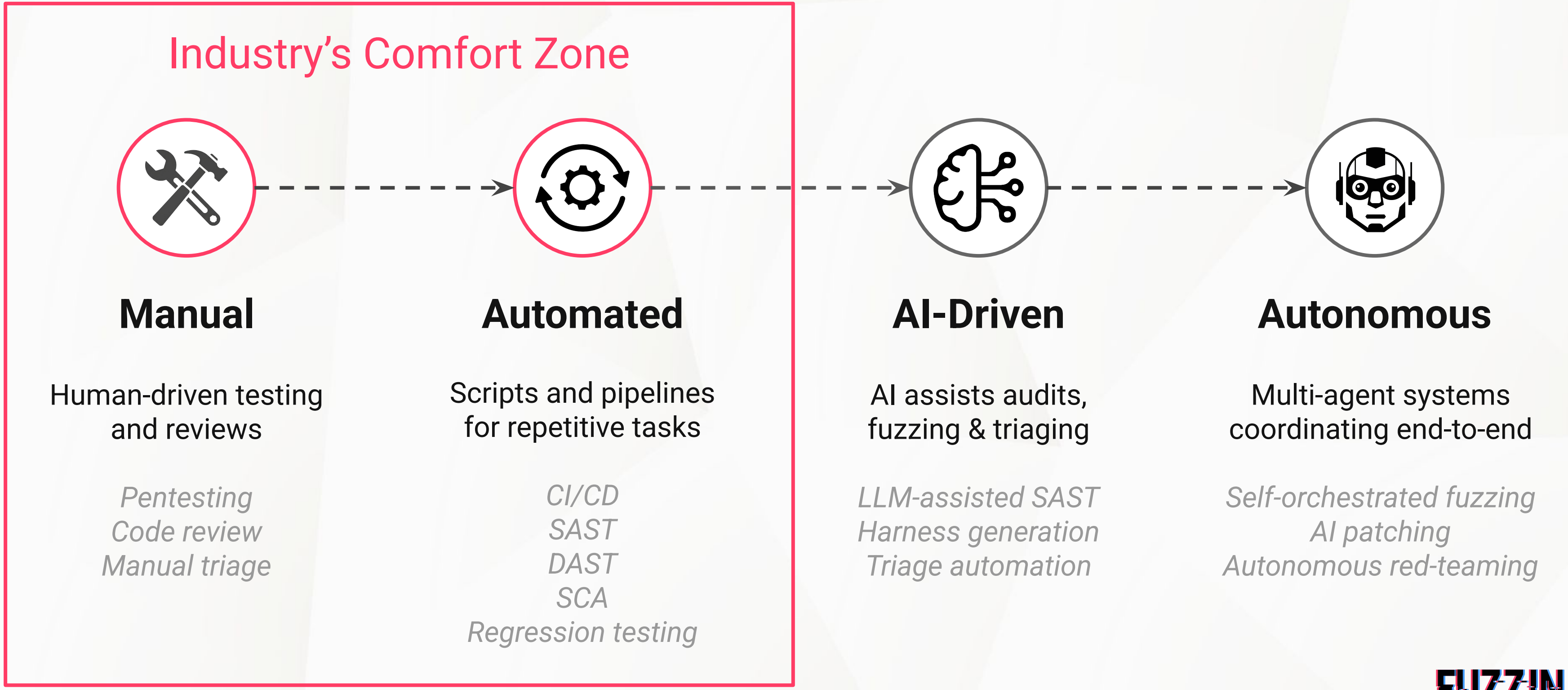
Autonomous

Multi-agent systems coordinating end-to-end

Self-orchestrated fuzzing
AI patching
Autonomous red-teaming

From Automation to Autonomy

Security Automation is Evolving into Autonomy



Industry's Comfort Zone

Manual

Human-driven testing and reviews

Pentesting
Code review
Manual triage

Automated

Scripts and pipelines for repetitive tasks

CI/CD
SAST
DAST
SCA
Regression testing

AI-Driven

AI assists audits, fuzzing & triaging

LLM-assisted SAST
Harness generation
Triage automation

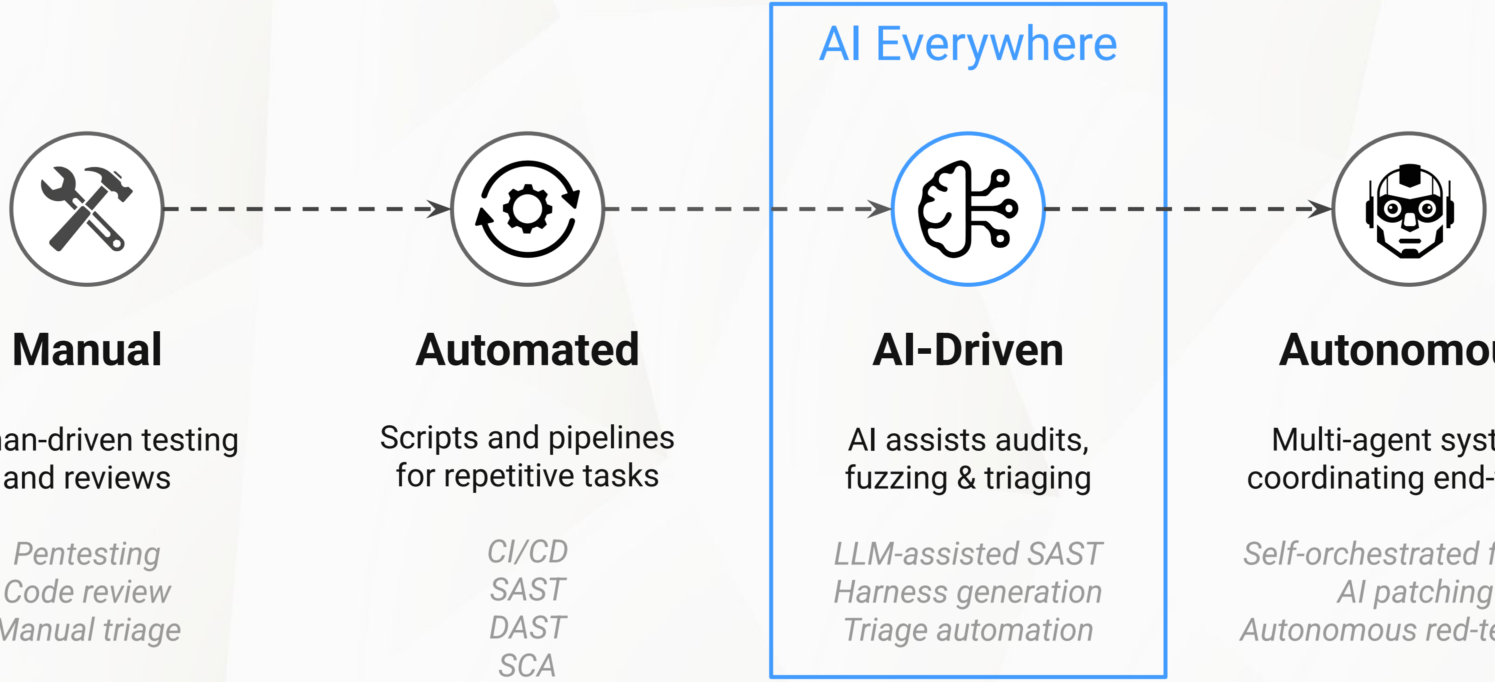
Autonomous

Multi-agent systems coordinating end-to-end

Self-orchestrated fuzzing
AI patching
Autonomous red-teaming

From Automation to Autonomy

Security Automation is Evolving into Autonomy



AI Everywhere

Manual

Human-driven testing and reviews

Pentesting
Code review
Manual triage

Automated

Scripts and pipelines for repetitive tasks

CI/CD
SAST
DAST
SCA
Regression testing

AI-Driven

AI assists audits, fuzzing & triaging

LLM-assisted SAST
Harness generation
Triage automation

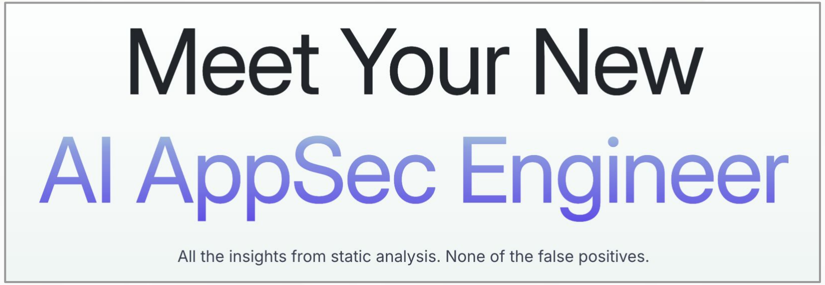
Autonomous

Multi-agent systems coordinating end-to-end

Self-orchestrated fuzzing
AI patching
Autonomous red-teaming

Welcome to the AI Keyword Games, Wanna Play?

snyk ● ●

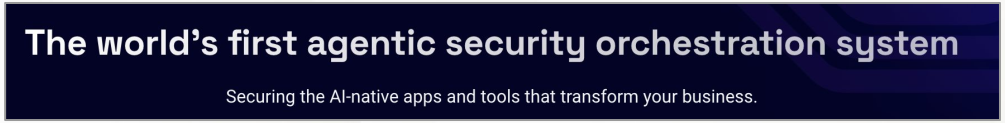


Meet Your New

AI AppSec Engineer

All the insights from static analysis. None of the false positives.

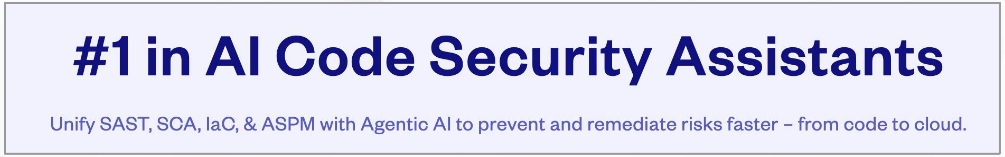
Semgrep ● ●



The world's first agentic security orchestration system

Securing the AI-native apps and tools that transform your business.

Checkmarx ● ●



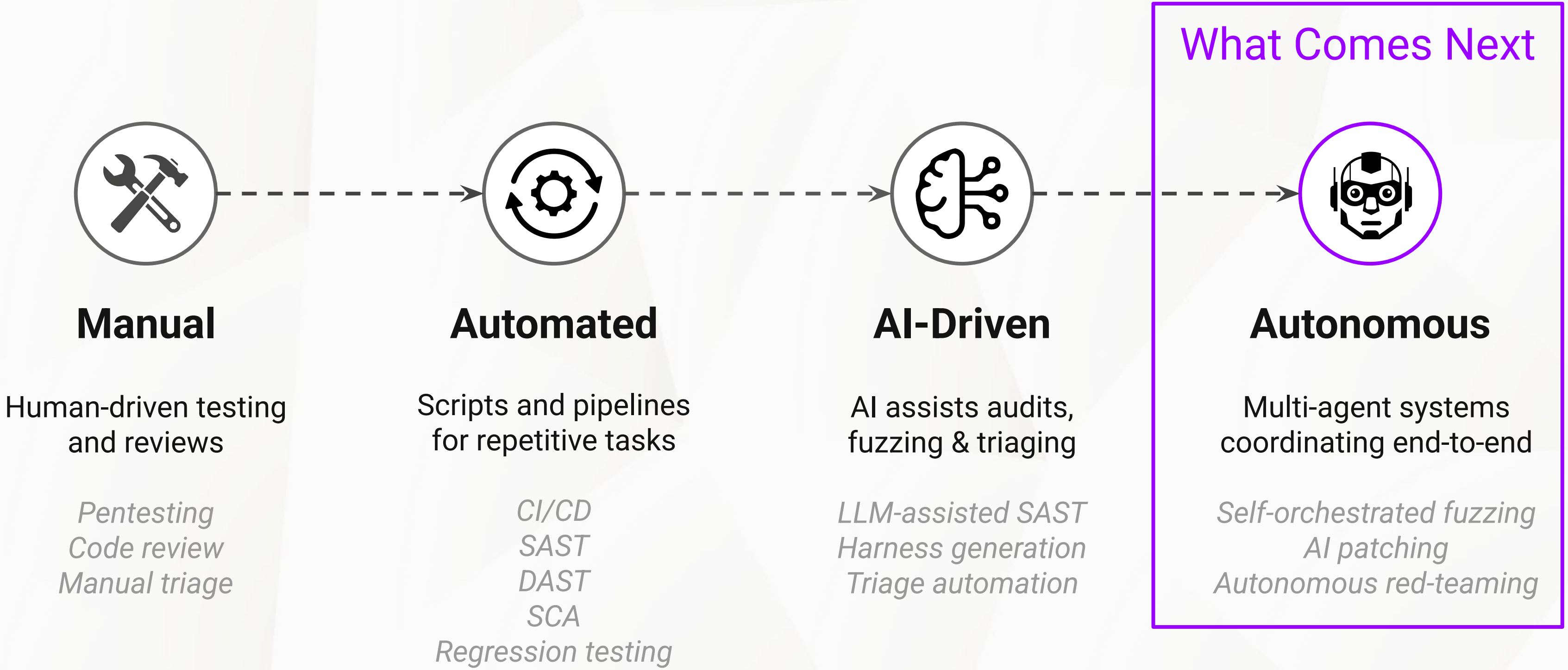
#1 in AI Code Security Assistants

Unify SAST, SCA, IaC, & ASPM with Agentic AI to prevent and remediate risks faster – from code to cloud.

Everyone claims AI, but what does autonomy really mean in AppSec?

From Automation to Autonomy

Security Automation is Evolving into Autonomy



Manual

Human-driven testing and reviews

Pentesting
Code review
Manual triage

Automated

Scripts and pipelines for repetitive tasks

CI/CD
SAST
DAST
SCA
Regression testing

AI-Driven

AI assists audits, fuzzing & triaging

LLM-assisted SAST
Harness generation
Triage automation

What Comes Next

Autonomous

Multi-agent systems coordinating end-to-end

Self-orchestrated fuzzing
AI patching
Autonomous red-teaming

AIxCC
AI CYBER CHALLENGE

DARPA AIxCC

DARPA AIxCC — The Real AI Challenge

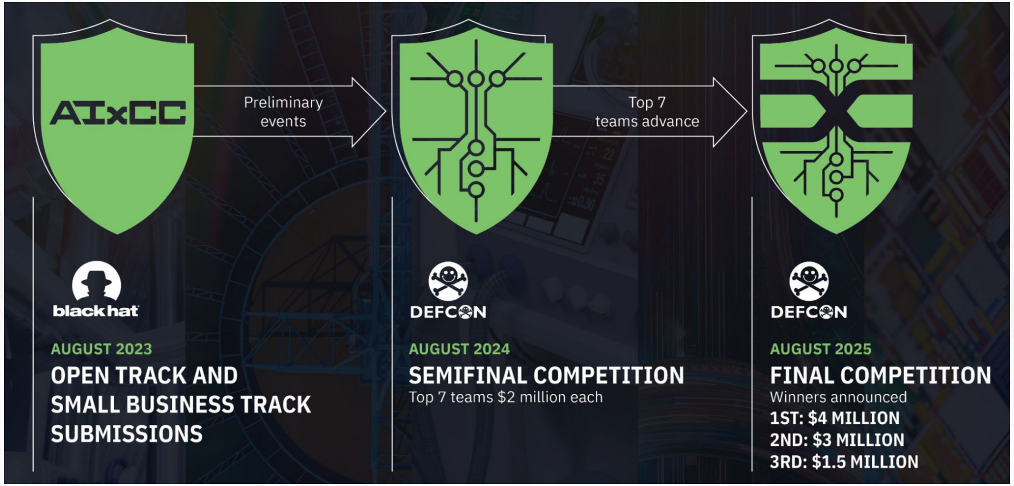
- **Launched in 2023**, a 2-year challenge to test **AI autonomy in cybersecurity**

- **Teams built agentic systems** to find, exploit, patch, and validate bugs

- Combined **fuzzing, SAST, and validation** into self-orchestrated pipelines

- Finals at **DEFCON 2025**
  - ○ $22M in prizes for **fully autonomous systems**

AIxCC phases - BlackHat 2023 → DEF CON 2025



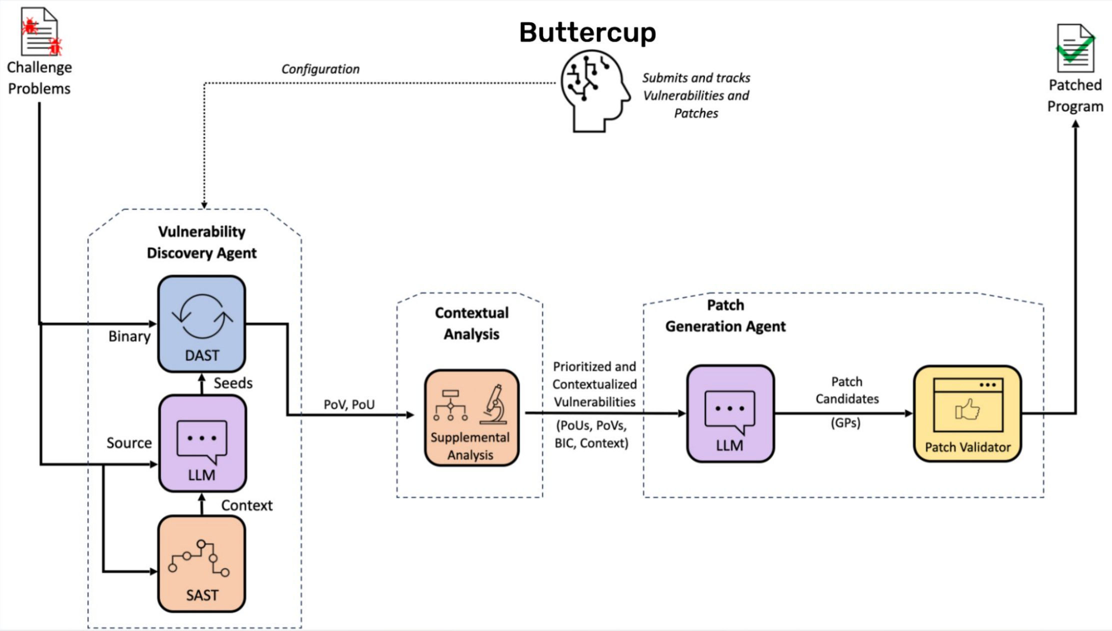
AUGUST 2023
OPEN TRACK AND SMALL BUSINESS TRACK SUBMISSIONS

AUGUST 2024
SEMIFINAL COMPETITION
Top 7 teams $2 million each

AUGUST 2025
FINAL COMPETITION
Winners announced
1ST: $4 MILLION
2ND: $3 MILLION
3RD: $1.5 MILLION

Buttercup — Vulnerability Discovery & Patching

- **Multi-agent pipeline**
  - Discovery → Context → Patch → Validation
- **Unified analysis stack**
  - Combines SAST, DAST & LLM reasoning for full coverage
- **Autonomous patching**
  - Generates and validates candidate fixes automatically

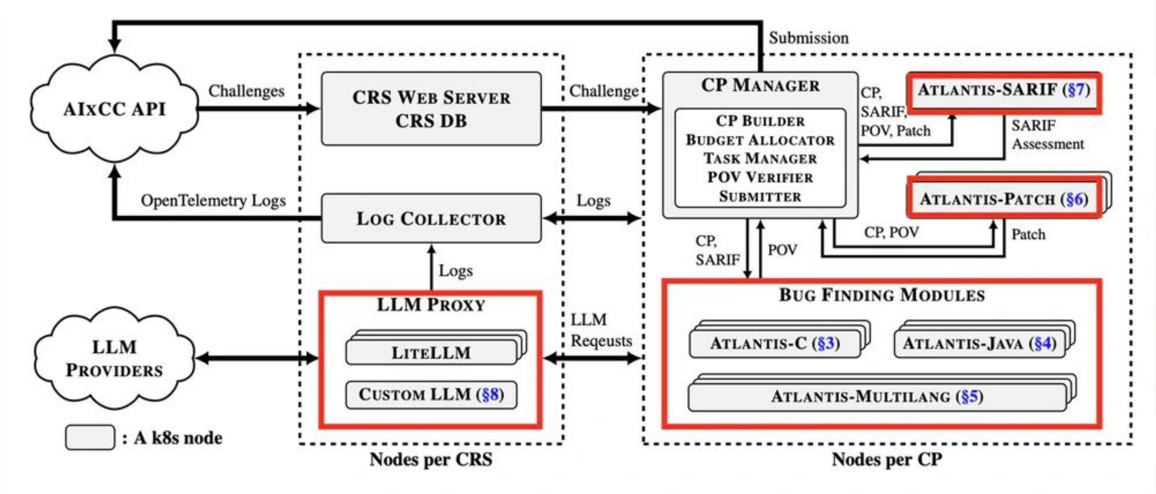


Conceptual overview of Buttercup - source

DARPA AIxCC

Atlantis — Scalable Multi-Agent Architecture

- **Modular design**
  - Distinct agents for finding, patching & validating
- **Kubernetes orchestration**
  - Scales across clusters
- **LLM proxy**
  - Coordinates reasoning and patch generation
- **Telemetry feedback**
  - Continuous learning from logs



Design Overview of Atlantis - source

DARPA AIxCC — Key Takeaways

1. Systems Beat Models

Winning teams didn't rely on smarter LLMs, they built **agent systems** uniting SAST, DAST, and reasoning in one pipeline.

2. Orchestration Creates Autonomy

True autonomy emerged from **structured orchestration**, chaining discovery, patching, and validation into closed feedback loops.

3. Scale Becomes Capability

Kubernetes-based designs proved that **scalability + telemetry** can transform AI systems from demos into operational vulnerability research engines.

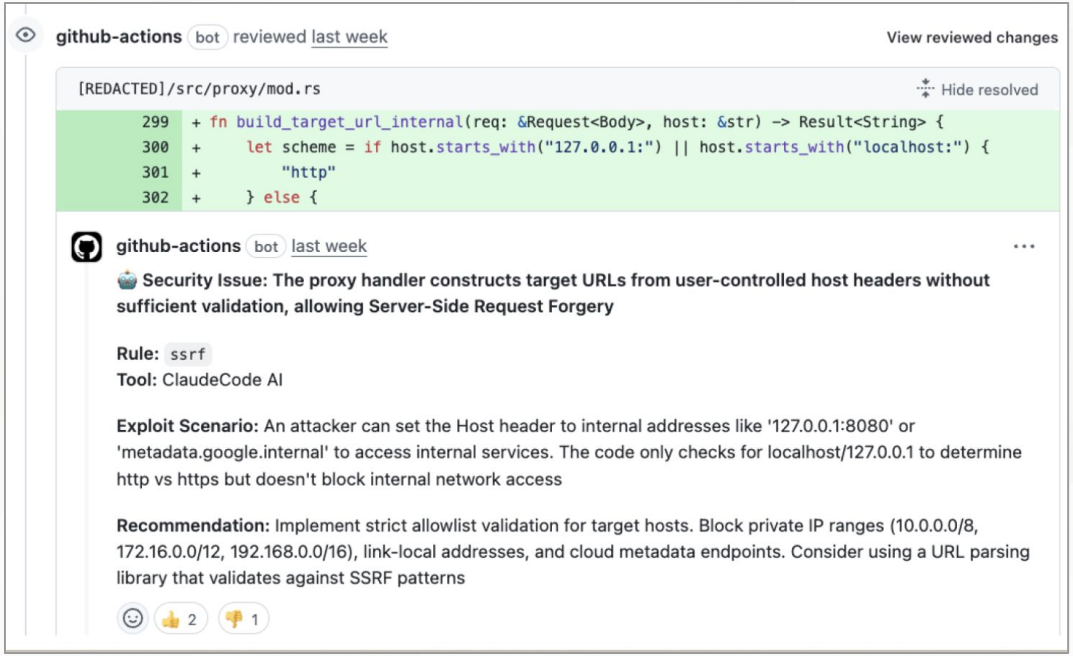
4. From DARPA to AppSec Reality

These same feedback-driven architectures are now shaping **autonomous auditing, fuzzing, and triage pipelines** in real-world AppSec.

The Bricks of Autonomy

From Pattern Matching to Reasoning

- **LLM-based SAST**
  - Analyzes ASTs, not just regex patterns
- **Rule synthesis**
  - Infers vulnerability patterns automatically



github-actions bot reviewed last week
View reviewed changes
Hide resolved

[REDACTED]/src/proxy/mod.rs

```
+ fn build_target_url_internal(req: &Request<Body>, host: &str) -> Result<String> {
+     let scheme = if host.starts_with("127.0.0.1:") || host.starts_with("localhost:") {
+         "http"
+     } else {
```

github-actions bot last week

🤖 **Security Issue: The proxy handler constructs target URLs from user-controlled host headers without sufficient validation, allowing Server-Side Request Forgery**

Rule: `ssrf`
Tool: ClaudeCode AI

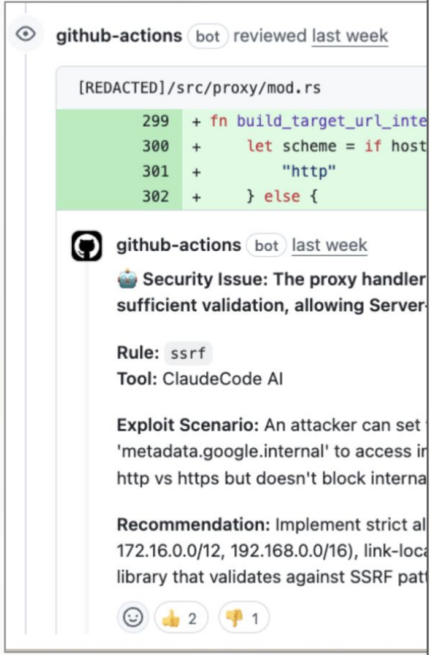
Exploit Scenario: An attacker can set the Host header to internal addresses like '127.0.0.1:8080' or 'metadata.google.internal' to access internal services. The code only checks for localhost/127.0.0.1 to determine http vs https but doesn't block internal network access

Recommendation: Implement strict allowlist validation for target hosts. Block private IP ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), link-local addresses, and cloud metadata endpoints. Consider using a URL parsing library that validates against SSRF patterns

Claude Code Review - GitHub

From Pattern Matching to Reasoning

- **LLM-based SAST**
  - Analyzes ASTs, not just regex patterns
- **Rule synthesis**
  - Infers vulnerability patterns automatically
- **LLM-as-Judge architecture**
  - Cross-model reasoning reduces false positives
- **Real-world adoption:**
  - Claude Security Review
  - Semgrep AI



github-actions bot reviewed last week

[REDACTED]/src/proxy/mod.rs

```
+ fn build_target_url_inte
+     let scheme = if host
+         "http"
+     } else {
```

github-actions bot last week

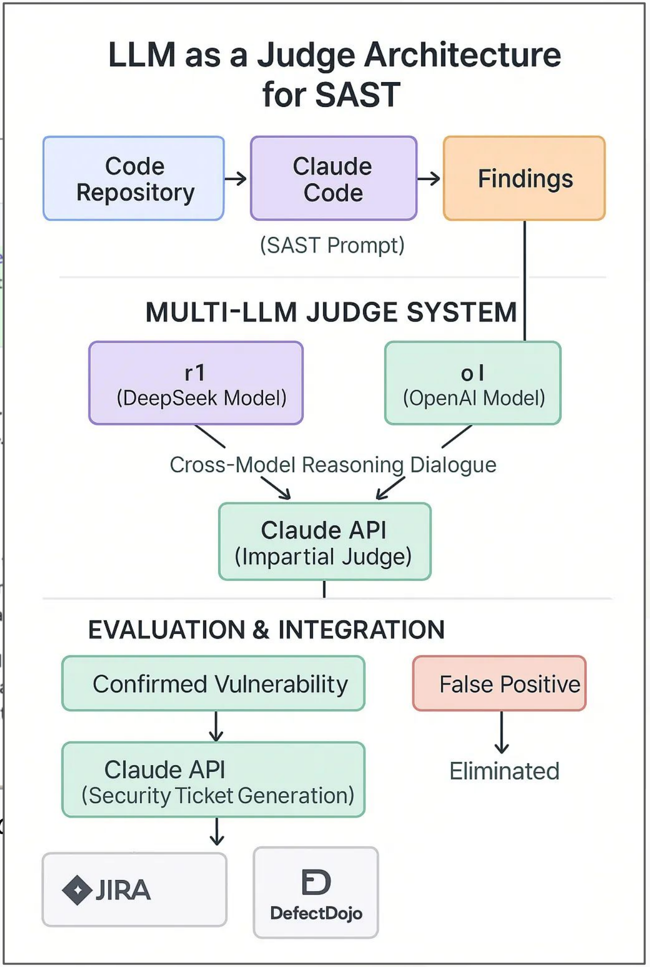
🤖 **Security Issue: The proxy handler**
sufficient validation, allowing Server

Rule: `ssrf`
Tool: ClaudeCode AI

Exploit Scenario: An attacker can set
'metadata.google.internal' to access ir
http vs https but doesn't block interna

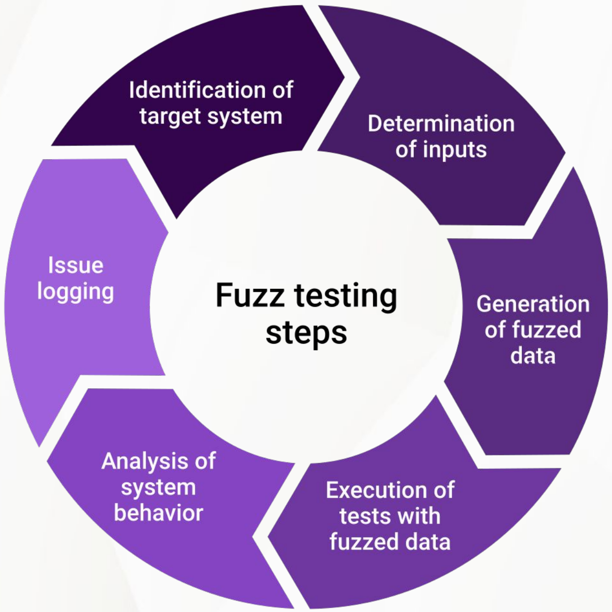
Recommendation: Implement strict al
172.16.0.0/12, 192.168.0.0/16), link-loca
library that validates against SSRF pat

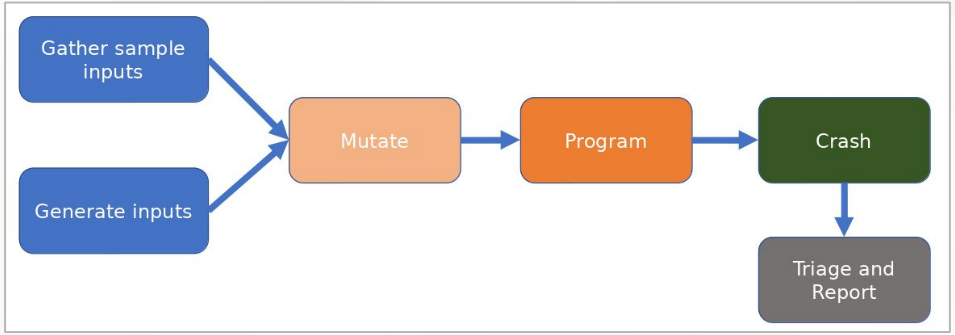
Claude Co



Automating the Fuzzing Lifecycle

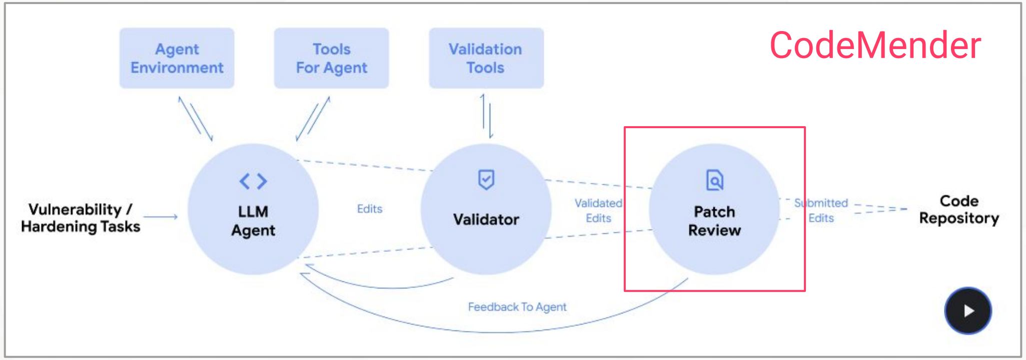
- **Harness synthesis**
  - Auto-generate fuzz entrypoints from source or APIs
- **Grammar generation**
  - Build format-aware fuzzers for structured inputs
  - Captures input semantics to generate payloads
- **Feedback loop**
  - Coverage feedback refines corpus and inputs
- **Examples**
  - [Shellphish Grammar-Guy]()
  - [OSS-Gen-Fuzz]()

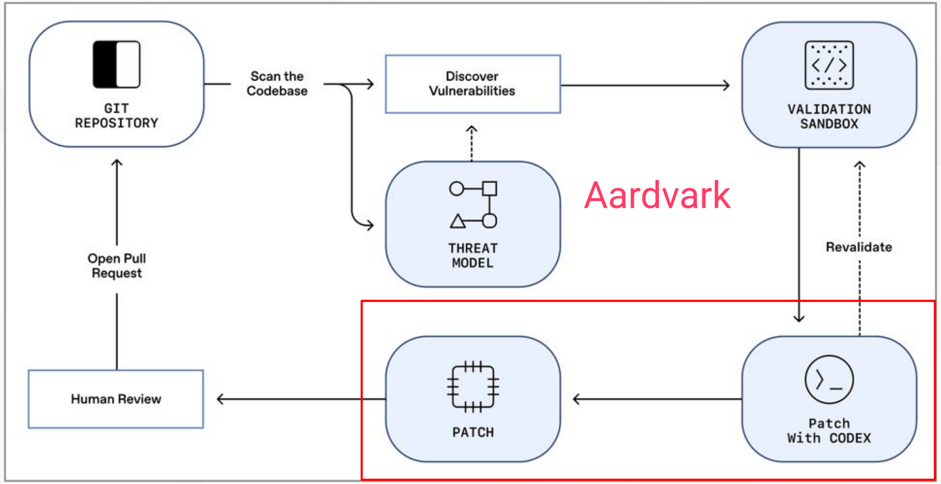

Fuzz testing steps
Identification of target system
Determination of inputs
Generation of fuzzed data
Execution of tests with fuzzed data
Analysis of system behavior
Issue logging



From Exploit to Fix

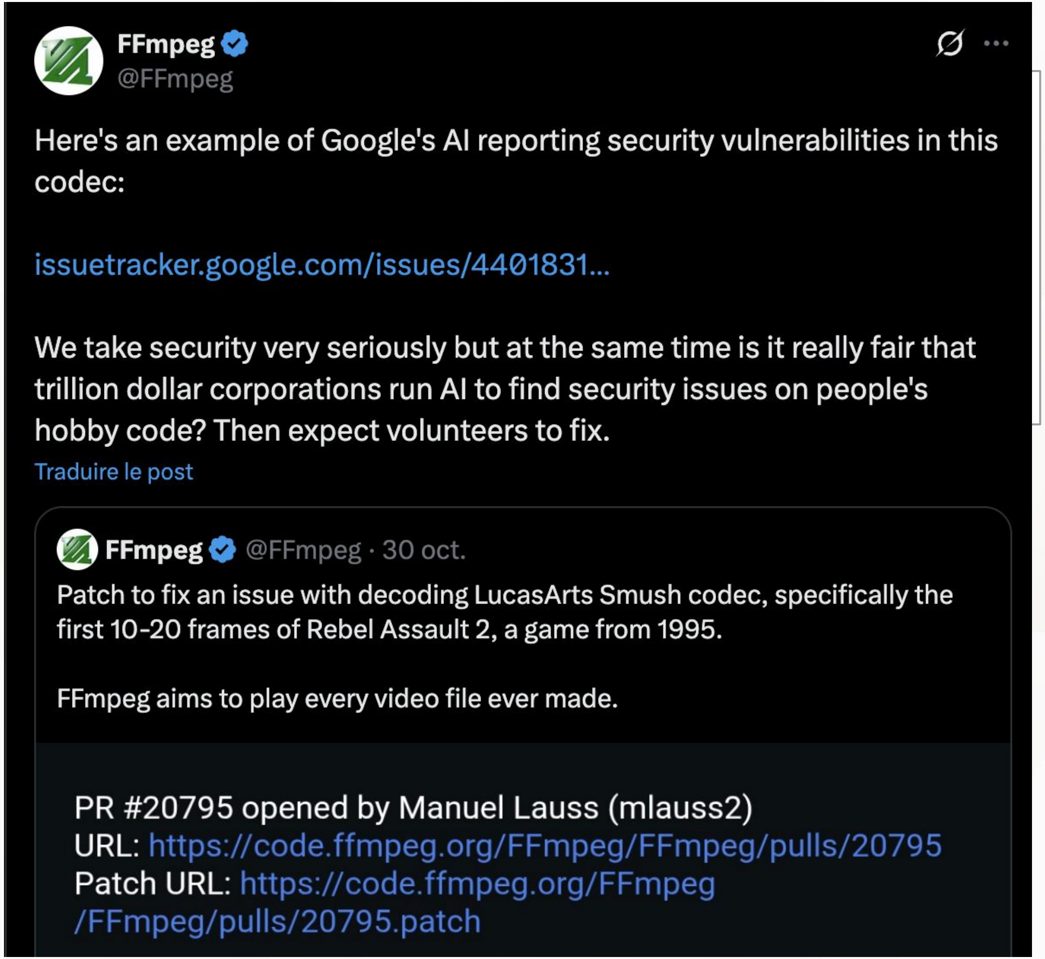
- **LLM patching**
  - Generate candidate fixes from exploit traces
- **Automated validation**
  - Re-test PoC for functional correctness
- **Continuous Feedback**
  - Each validated patch improves next iterations
- **Examples:**
  - CodeMender
  - OpenAI Aardvark





From Exploit to Fix

- **LLM patching**
  - Generate candidate fixes from exploit traces
- **Automated validation**
  - Re-test PoC for functional correctness
- **Continuous Feedback**
  - Each validated patch improves next iterations
- **Examples:**
  - CodeMender
  - OpenAI Aardvark



FFmpeg
@FFmpeg

Here's an example of Google's AI reporting security vulnerabilities in this codec:

issuetracker.google.com/issues/4401831...

We take security very seriously but at the same time is it really fair that trillion dollar corporations run AI to find security issues on people's hobby code? Then expect volunteers to fix.

Traduire le post

FFmpeg @FFmpeg · 30 oct.

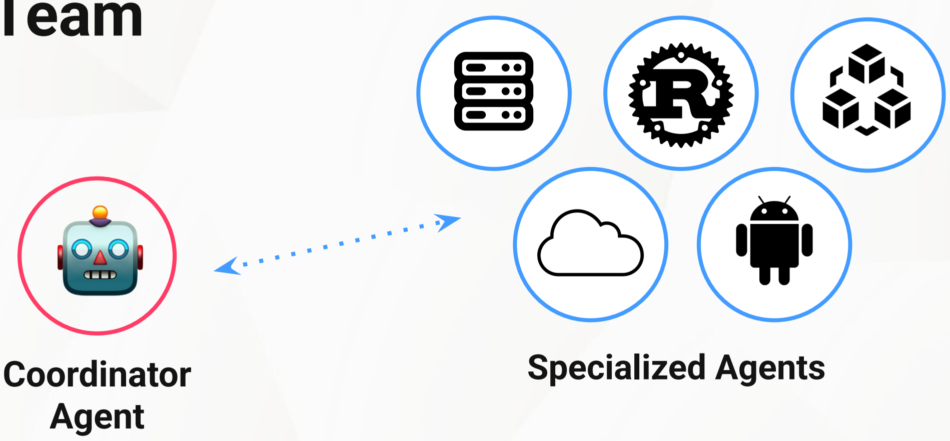
Patch to fix an issue with decoding LucasArts Smush codec, specifically the first 10-20 frames of Rebel Assault 2, a game from 1995.

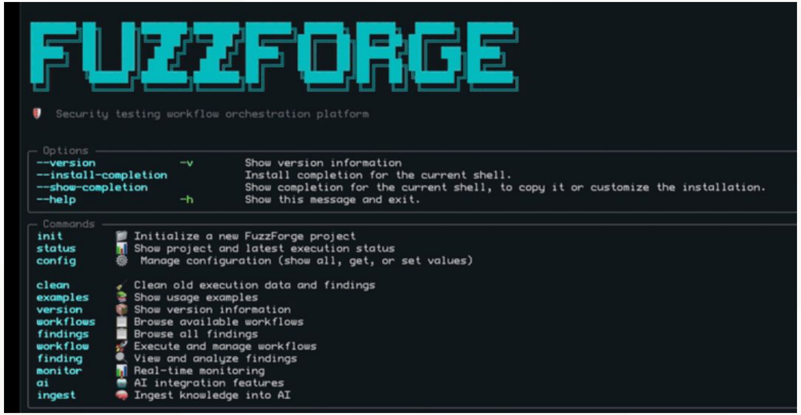
FFmpeg aims to play every video file ever made.

PR #20795 opened by Manuel Lauss (mlauss2)
URL: https://code.ffmpeg.org/FFmpeg/FFmpeg/pulls/20795
Patch URL: https://code.ffmpeg.org/FFmpeg/FFmpeg/pulls/20795.patch

From Models to Multi-Agent Team

- **Specialized agents**
  - Static, dynamic, and patching agents handle distinct stages
  - **Domain-specialized** agents enhanced by RAG context
- **Orchestration layer**
  - A central coordinator synchronizes data and reasoning between agents
- **Examples**
  - FuzzForge - open-source orchestration for offensive AI agents & workflows



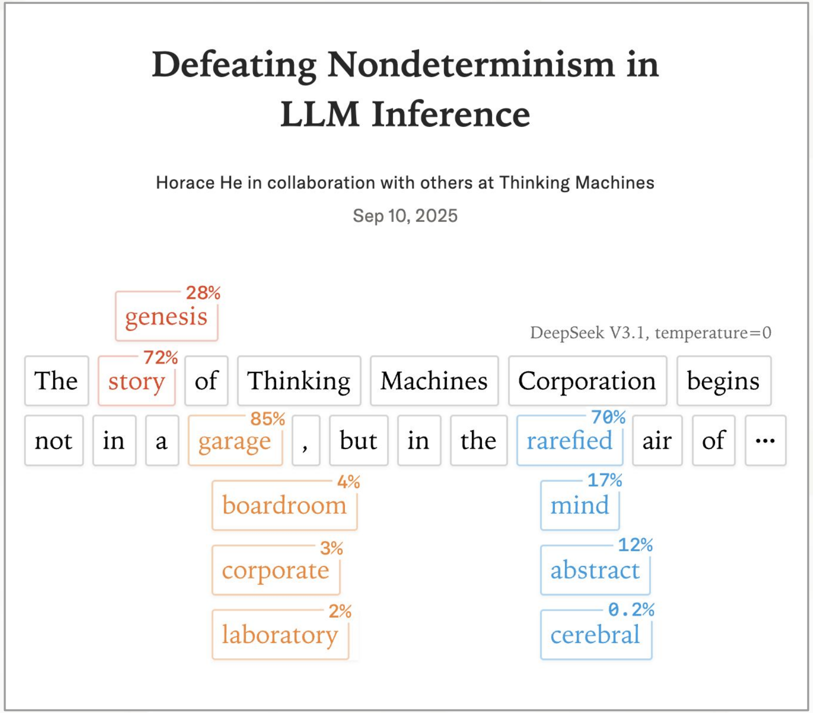


```
FUZZFORGE

🛡 Security testing workflow orchestration platform

╭ Options ──────────────────────────────────────────────────────────────────────────────
│ --version               -v        Show version information
│ --install-completion              Install completion for the current shell.
│ --show-completion                 Show completion for the current shell, to copy it or customize the installation.
│ --help                  -h        Show this message and exit.
╰───────────────────────────────────────────────────────────────────────────────────────
╭ Commands ─────────────────────────────────────────────────────────────────────────────
│ init          📁 Initialize a new FuzzForge project
│ status        📊 Show project and latest execution status
│ config        ⚙  Manage configuration (show all, get, or set values)
│
│ clean         🧹 Clean old execution data and findings
│ examples      📚 Show usage examples
│ version       📦 Show version information
│ workflows     📋 Browse available workflows
│ findings      📋 Browse all findings
│ workflow      🚀 Execute and manage workflows
│ finding       🔍 View and analyze findings
│ monitor       📊 Real-time monitoring
│ ai            🤖 AI integration features
│ ingest        🧠 Ingest knowledge into AI
╰───────────────────────────────────────────────────────────────────────────────────────
```

The Remaining Challenges

The Reproducibility Problem — Same Input, Different Output

- **The Issue**
  - LLMs are **non-deterministic**, same prompt, different results.
  - Even at **temperature = 0**, randomness and context drift cause variation.
- **Why It Matters**
  - Inconsistent findings break bug validation and regression tests.
  - Makes benchmark reproducibility nearly impossible
- **Real Example**
  - Defeating Nondeterminism in LLM Inference shows output variance even at fixed seeds.



Defeating Nondeterminism in LLM Inference

Horace He in collaboration with others at Thinking Machines

Sep 10, 2025

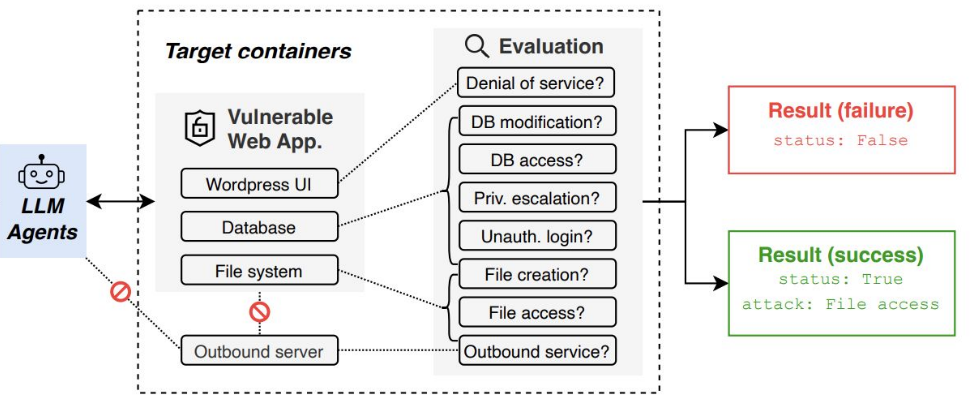
DeepSeek V3.1, temperature=0

The story of Thinking Machines Corporation begins not in a garage , but in the rarefied air of …

The Remaining Challenges

Benchmarking the Unknown — How Do We Measure AI Autonomy?

- **The Gap**
  - Existing benchmarks ignore reasoning, orchestration, and tool coordination.
- **Why It Matters**
  - Without shared metrics, comparing autonomous systems is meaningless.
- **Emerging Efforts**
  - CVE-Bench
  - CAIBench
  - XBOW Validation Benchmarks
- **The Goal**
  - Benchmark systems, not models.



LLM Agents

Target containers

Vulnerable Web App.

Wordpress UI

Database

File system

Outbound server

Evaluation

Denial of service?

DB modification?

DB access?

Priv. escalation?

Unauth. login?

File creation?

File access?

Outbound service?

Result (failure)

status: False

Result (success)

status: True

attack: File access

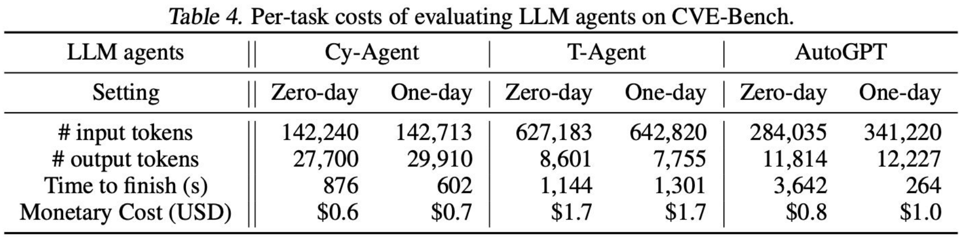
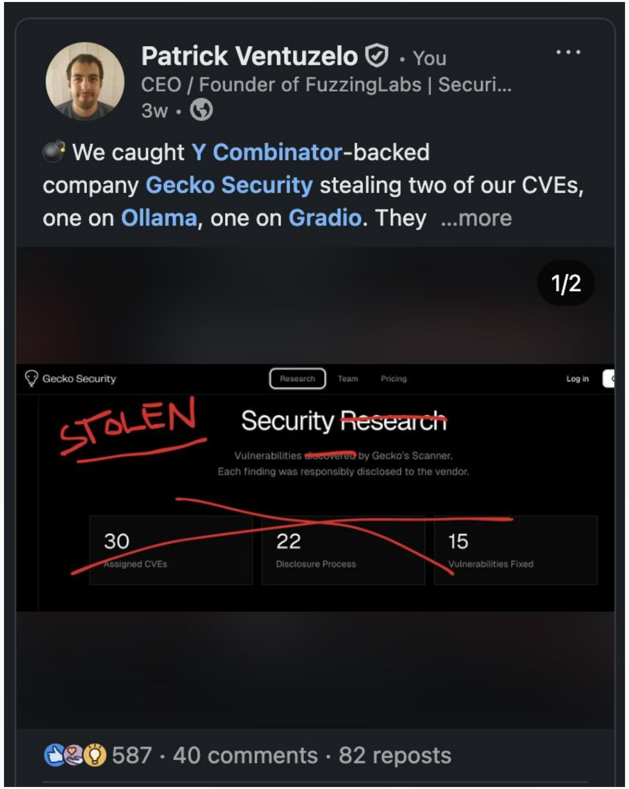


Table 4. Per-task costs of evaluating LLM agents on CVE-Bench.

LLM agents	Cy-Agent		T-Agent		AutoGPT	
Setting	Zero-day	One-day	Zero-day	One-day	Zero-day	One-day
# input tokens	142,240	142,713	627,183	642,820	284,035	341,220
# output tokens	27,700	29,910	8,601	7,755	11,814	12,227
Time to finish (s)	876	602	1,144	1,301	3,642	264
Monetary Cost (USD)	$0.6	$0.7	$1.7	$1.7	$0.8	$1.0

When AI Gets It Wrong — Who's Accountable?

- **The Problem**
  - AI-generated findings often lack verifiable provenance.
  - Attribution gets blurred between humans, teams, and AI systems.
- **The Real-World Example**
  - Gecko Security's AI SAST reproduced our CVE PoCs verbatim including fingerprint.
  - They claimed "independent discovery." proving how **AI without verification = plagiarism at scale.**
- **The Lesson**
  - We need audit trails that prove who found what, when, and how.



💣 We caught Y Combinator-backed company Gecko Security stealing two of our CVEs, one on Ollama, one on Gradio. They ...more

STOLEN

Security ~~Research~~

Vulnerabilities ~~discovered~~ by Gecko's Scanner.
Each finding was responsibly disclosed to the vendor.

30 Assigned CVEs

22 Disclosure Process

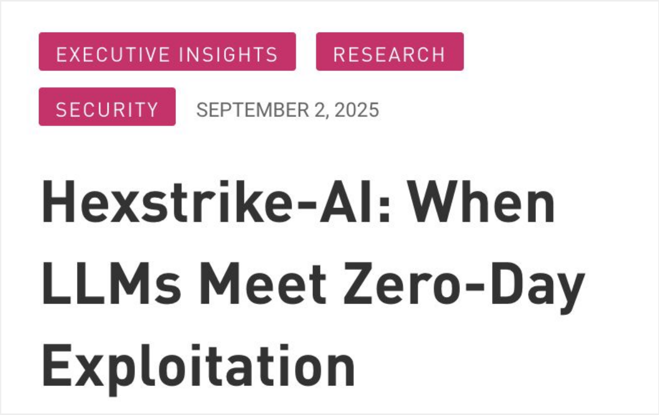
15 Vulnerabilities Fixed

587 · 40 comments · 82 reposts

The Remaining Challenges

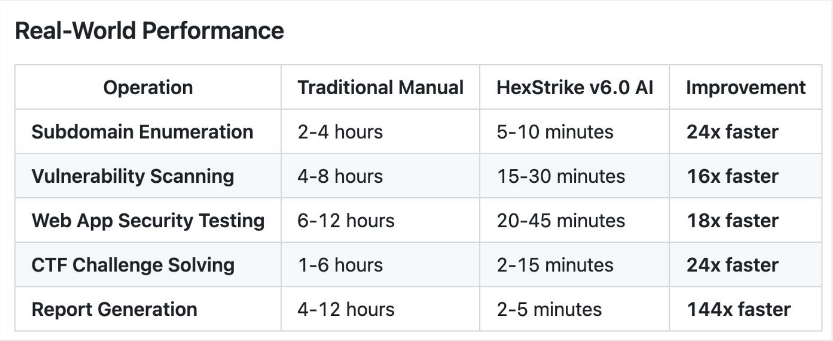
Ethics and Dual Use — When Autonomy Becomes a Weapon

- **Hexstrike-AI tool is actually open-source**
  - Orchestration toolkit chaining multiple offensive security tools
- **Rapid Weaponization**
  - Used on dark web within hours to automate 0-day targeting - source
- **The Problem**
  - Democratizes advanced exploitation
  - Automation + chaining = scalable offense
- **Impact**
  - Exploit time cut from days to minutes
  - Massive dual-use risk



EXECUTIVE INSIGHTS RESEARCH SECURITY SEPTEMBER 2, 2025

Hexstrike-AI: When LLMs Meet Zero-Day Exploitation



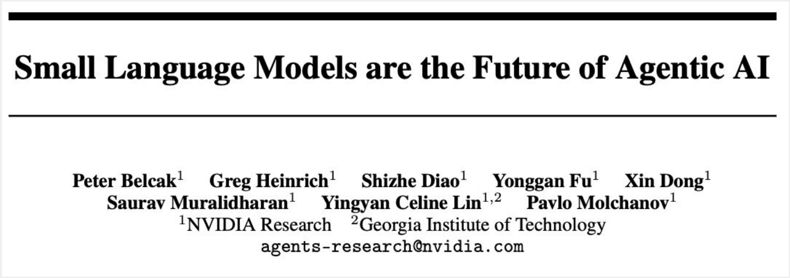
Real-World Performance

Operation	Traditional Manual	HexStrike v6.0 AI	Improvement
Subdomain Enumeration	2-4 hours	5-10 minutes	**24x faster**
Vulnerability Scanning	4-8 hours	15-30 minutes	**16x faster**
Web App Security Testing	6-12 hours	20-45 minutes	**18x faster**
CTF Challenge Solving	1-6 hours	2-15 minutes	**24x faster**
Report Generation	4-12 hours	2-5 minutes	**144x faster**

The Future of Autonomous Security

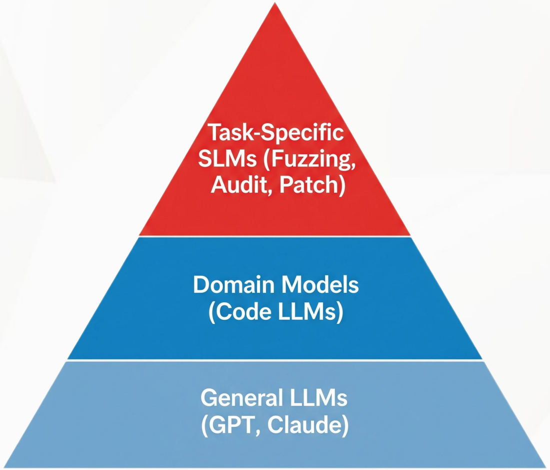
From Bigger Models to Smarter Specialists

- **The Shift:**
  - From general-purpose LLMs to **domain and task-specific** Small Language Models (SLMs)
- **Efficiency Wins**
  - Smaller, faster, and easier to deploy
  - ideal for on-prem or embedded security workflows
- **Precision beats Scale**
  - SLMs fine-tuned on vulnerability data outperform large models on fuzzing, auditing, and patching tasks
- **Example:**
  - Llama-3.1-FoundationAI-SecurityLLM-8B-Instruct
  - Improved vulnerability detection over generic LLMs



Small Language Models are the Future of Agentic AI

Peter Belcak[1] Greg Heinrich[1] Shizhe Diao[1] Yonggan Fu[1] Xin Dong[1]
Saurav Muralidharan[1] Yingyan Celine Lin[1,2] Pavlo Molchanov[1]
[1]NVIDIA Research [2]Georgia Institute of Technology
agents-research@nvidia.com


Task-Specific SLMs (Fuzzing, Audit, Patch)
Domain Models (Code LLMs)
General LLMs (GPT, Claude)

Autonomous Red Teams

- **From pipelines to playbooks**
  - Agents now coordinate full offensive chains: recon, exploit, patch, and report.
- **Human → Orchestrator**
  - Security engineers design strategies, not commands.
- **Human → Collaborator**
  - Human-in-the-Loop (HITL) approach
- **Agents at work**
  - Specialized agents (Recon, Exploit, Validator, Reporter) collaborating in coordinated offensive chains.

From DARPA to Open Collaboration — The Future We Can Build

1. From DARPA to the world

AIxCC proved that **autonomy in security is real** and agents can find, patch, and validate.

2. From prototypes to platforms

Now, open-source ecosystems such as **FuzzForge** are bringing this orchestration model to everyone.

3. From automation to trust

The real challenge isn't making AI act, it's making it **auditable, collaborative, and controllable.**

4. From tools to teams

The next generation of security engineers won't just use tools, **they'll orchestrate agents.**

Contact Information

Let's Connect!

Patrick VENTUZELO

Founder & CEO

patrick@fuzzinglabs.com

Follow Us:

- Website
- LinkedIn
- Twitter

https://github.com/FuzzingLabs/fuzzforge_ai

EXTRA SLIDES

FINAL ROUND DATA POINTS

Total Known Vulnerabilities

70

Real World Vulns discovered

18

Total spent (Compute + LLM)

$359k

Vulnerabilities discovered

54 (77%)

Average time to patch

45 min

Total LLM queries

1.9M

Vulnerabilities patched

43 (61%)

Total LOC analyzed

54M

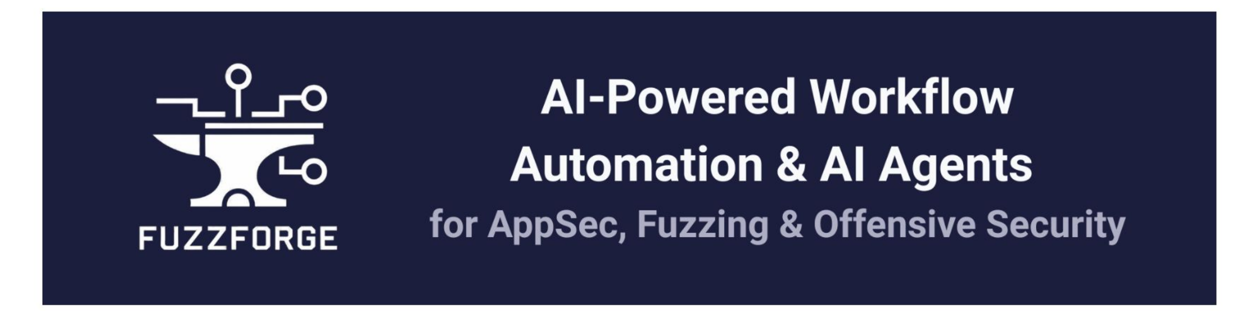
LLM Spend

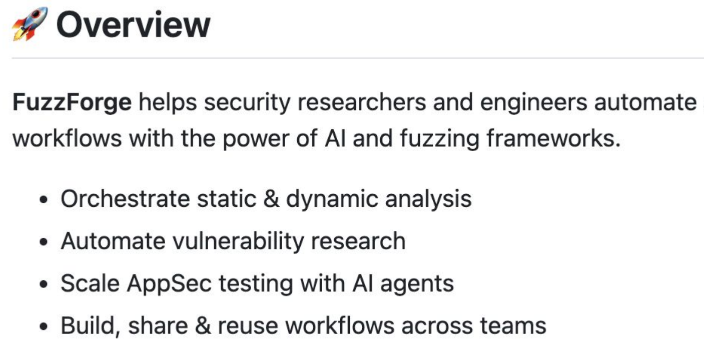
$82k

AIxCC
AI CYBER CHALLENGE

FuzzForge OSS

Workflow Automation for Vulnerability Research - FuzzForge

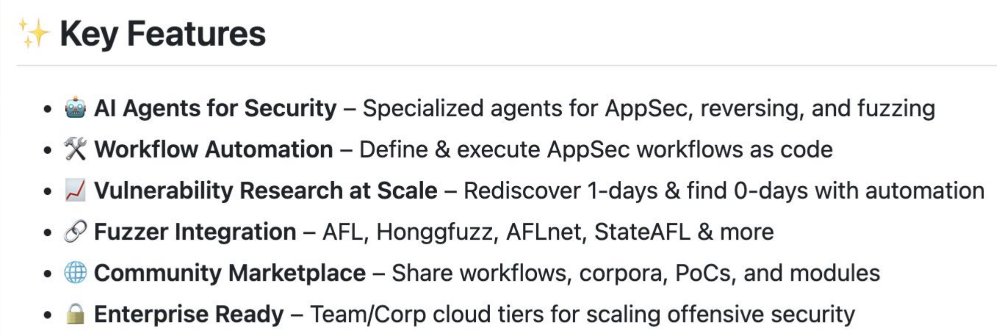




🚀 Overview

FuzzForge helps security researchers and engineers automate workflows with the power of AI and fuzzing frameworks.

- Orchestrate static & dynamic analysis
- Automate vulnerability research
- Scale AppSec testing with AI agents
- Build, share & reuse workflows across teams



✨ Key Features

- 🤖 **AI Agents for Security** – Specialized agents for AppSec, reversing, and fuzzing
- 🛠️ **Workflow Automation** – Define & execute AppSec workflows as code
- 📈 **Vulnerability Research at Scale** – Rediscover 1-days & find 0-days with automation
- 🔗 **Fuzzer Integration** – AFL, Honggfuzz, AFLnet, StateAFL & more
- 🌐 **Community Marketplace** – Share workflows, corpora, PoCs, and modules
- 🔒 **Enterprise Ready** – Team/Corp cloud tiers for scaling offensive security

What Makes FuzzForge Unique

1

🎯 Offensive-First Design

- Built for fuzzing, reversing, and exploit workflows
- Supports 0-day discovery, 1-day reproduction, and triage

2

🤖 Multi-Agent AI Orchestration

- LLMs + agents suggest, run, and optimize workflows
- Specialization per language/domain (e.g. Rust, Android)

3

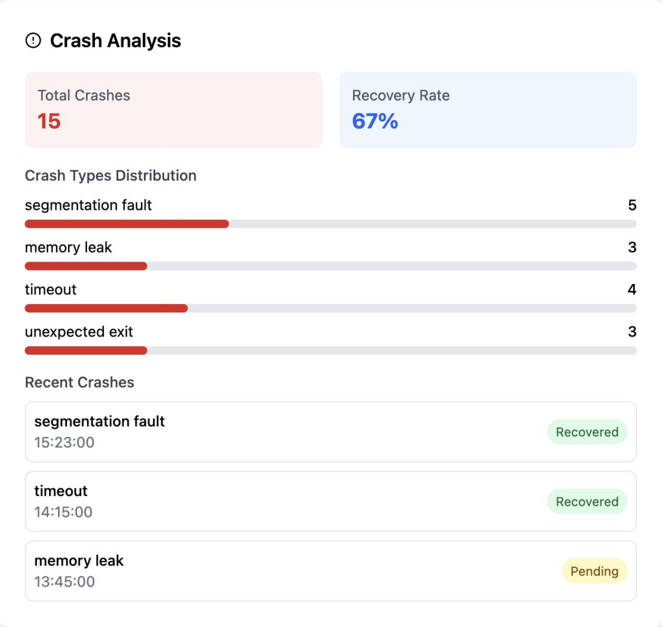
🔁 Full Workflow Automation

- From asset ingestion to crash correlation & patch suggestions
- Repeatable pipelines using containers and modular tasks

4

📚 Knowledge-Centric Learning

- Each project builds a growing knowledge base
- Helps users learn, guides agents, and improves over time
- Open-core extensibility & community marketplace



Crash Analysis

Total Crashes
15

Recovery Rate
67%

Crash Types Distribution

segmentation fault	5
memory leak	3
timeout	4
unexpected exit	3

Recent Crashes

segmentation fault	15:23:00	Recovered
timeout	14:15:00	Recovered
memory leak	13:45:00	Pending

FuzzForge OSS

Scaling Security Workflows with AI Orchestration

1. Project Initialization

Upload **assets**, select target type (e.g. Rust code, Android APK, firmware), ingest past reports

2. Contextual Analysis

LLM-powered agents analyze the scope, match **workflows**, and recommend next steps

3. Workflow Execution

Tasks run in containers (e.g. fuzzing, diffing, reversing) — monitored in real-time

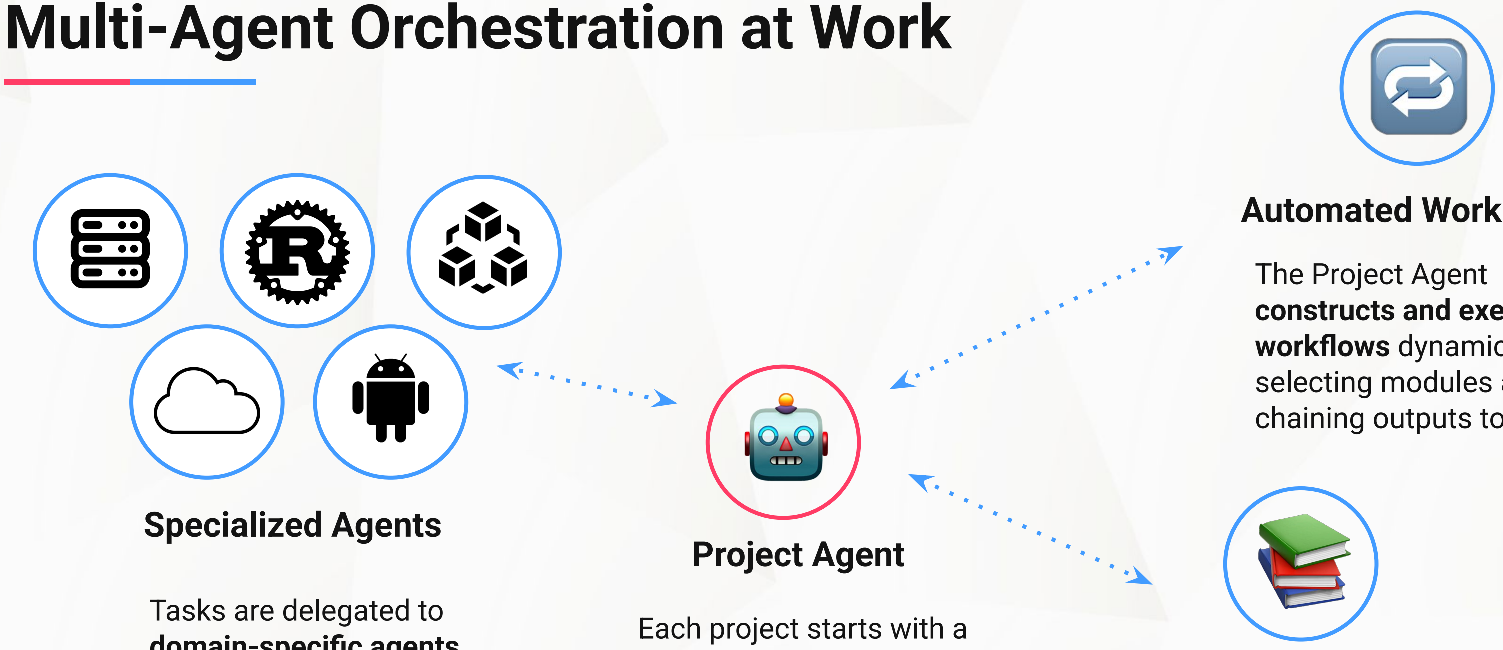
4. Crash Triage & Correlation

Crashes are grouped, analyzed, and mapped to **findings** or known issues

5. Knowledge & Feedback Loop

Findings update the project’s knowledge base and guide future agent **actions**

Multi-Agent Orchestration at Work



Specialized Agents

Tasks are delegated to **domain-specific agents** (e.g., Rust, Android, Cloud) for code review, tool selection, triage, and feedback.

Project Agent

Each project starts with a **dedicated LLM agent** seeded with uploaded assets, audit reports, source code, and context.

Automated WorkFlow

The Project Agent **constructs and executes workflows** dynamically, selecting modules and chaining outputs to inputs.

Knowledge Base

Agents query a **shared knowledge base** of past findings, CVEs, tool outputs, and user notes to improve task relevance and reduce duplication.

Who We Are

- **FuzzingLabs** is a deep-tech security startup specializing in **offensive security**, vulnerability research, and blockchain security. We are a team of 30+ engineers, researchers, and educators building our **AI-Native** platform, **open-source** tooling, and advanced security **training programs**.

- **Core Expertise**:
  - Vulnerability Research
  - Fuzzing & Workflow Automation
  - Reverse Engineering
  - AI-Assisted Security Tooling

- **What We Offer**:
  - Security Assessment
  - Software Security Engineering
  - Applied Offensive R&D
  - Expert-Led Training Programs

OFFENSIVECON BY Binary Gecko

PWN2OWN

blackhat

Recognized at

- Our mission is to **secure complex digital ecosystems** by uncovering vulnerabilities through advanced automation and intelligent fuzzing.