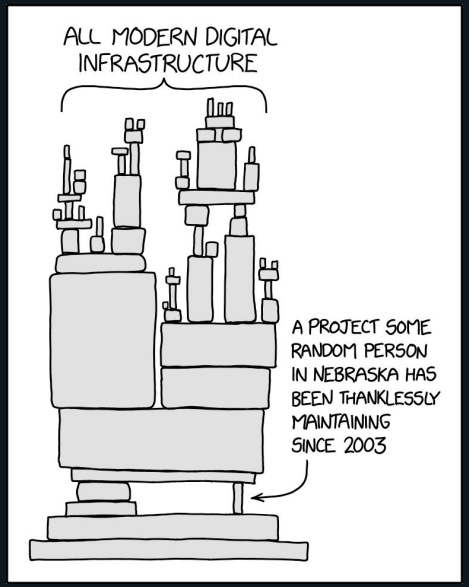
10/23/2025

Buttercup: Autonomously Finding and Fixing Bugs at Scale in Open-Source Software

State of Open Source Security

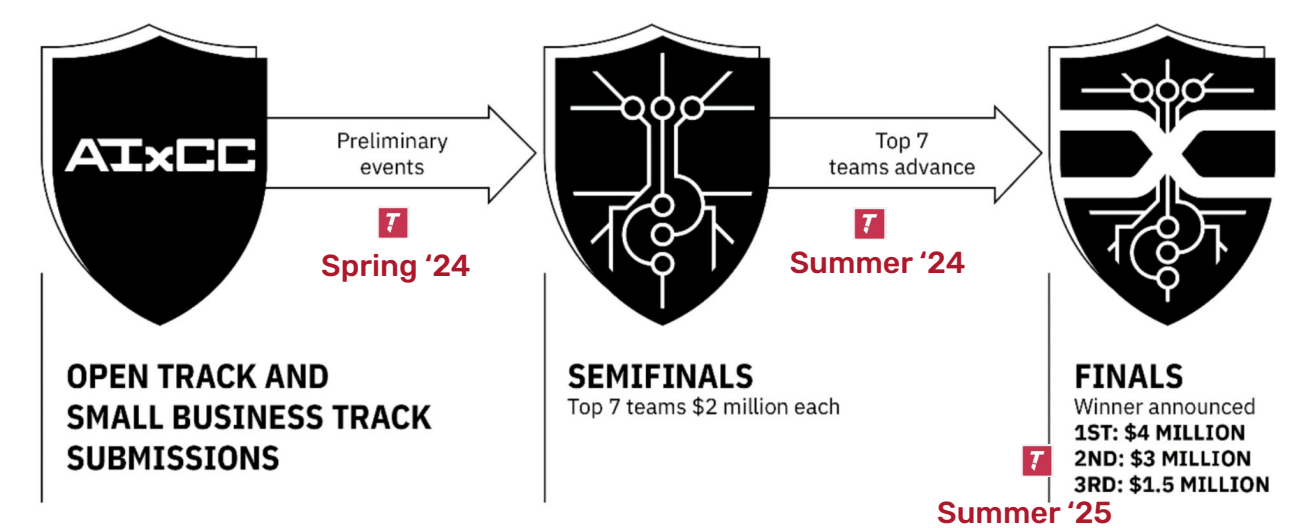


Source: xkcd

- **World runs on critical open-source projects maintained by small teams or individuals**
- **Bugs in these projects are everyone's problem**
  - **Log4j**
- **Maintainers already have too many open issues and bug reports**
- **More bug reports won't help them!**

AI Cyber Challenge (AIxCC)

Competition to create AI systems that find and fix bugs in open-source software



Competition Rules

- **Fully automated solution: no human in the loop**
- **Points for:**
  - Patches - Worth the most points
  - Vulnerabilities - Requires input which triggers bug
  - Static analysis alerts - Minor points
  - Bundling - Match patches, vulnerabilities, and alerts
- **Scoring modifiers:**
  - Speed - Earlier submissions get more points
  - Accuracy - Incorrect/duplicate submissions reduce points
- **Penalties**
  - Duplicate PoVs or patches reduce Accuracy Multiplier
  - Incorrect patches, including patches that fail with a new PoV

How would you solve this?

Competing in the Finals

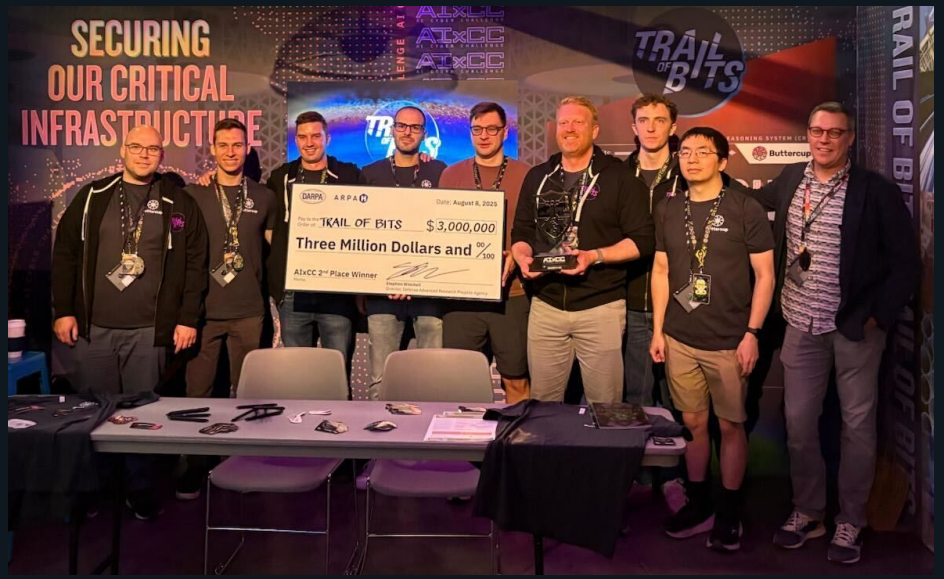
- **Adapting to finals:**
  - Semifinals: $100 LLM budget per challenge
  - Finals: $50,000 LLM budget and $85,000 Azure budget
- **Race to implement features in time for practice rounds**
  - Three practice rounds
  - Each round is an opportunity to find bugs
- **Last 2 months were especially challenging**
  - Balanced shipping improvements with stability
  - Constant testing

Results: 2nd Place and $3 Million

- **Buttercup won 2nd place and a $3 Million prize**
- 90%+ Accuracy
- Found 28 vulnerabilities across 20 CWE Categories
- Patched 19 vulnerabilities

Keys to success:

- Accuracy
- Performing well across tasks
- High reliability



How did Buttercup win 2nd place?

Approach behind Buttercup

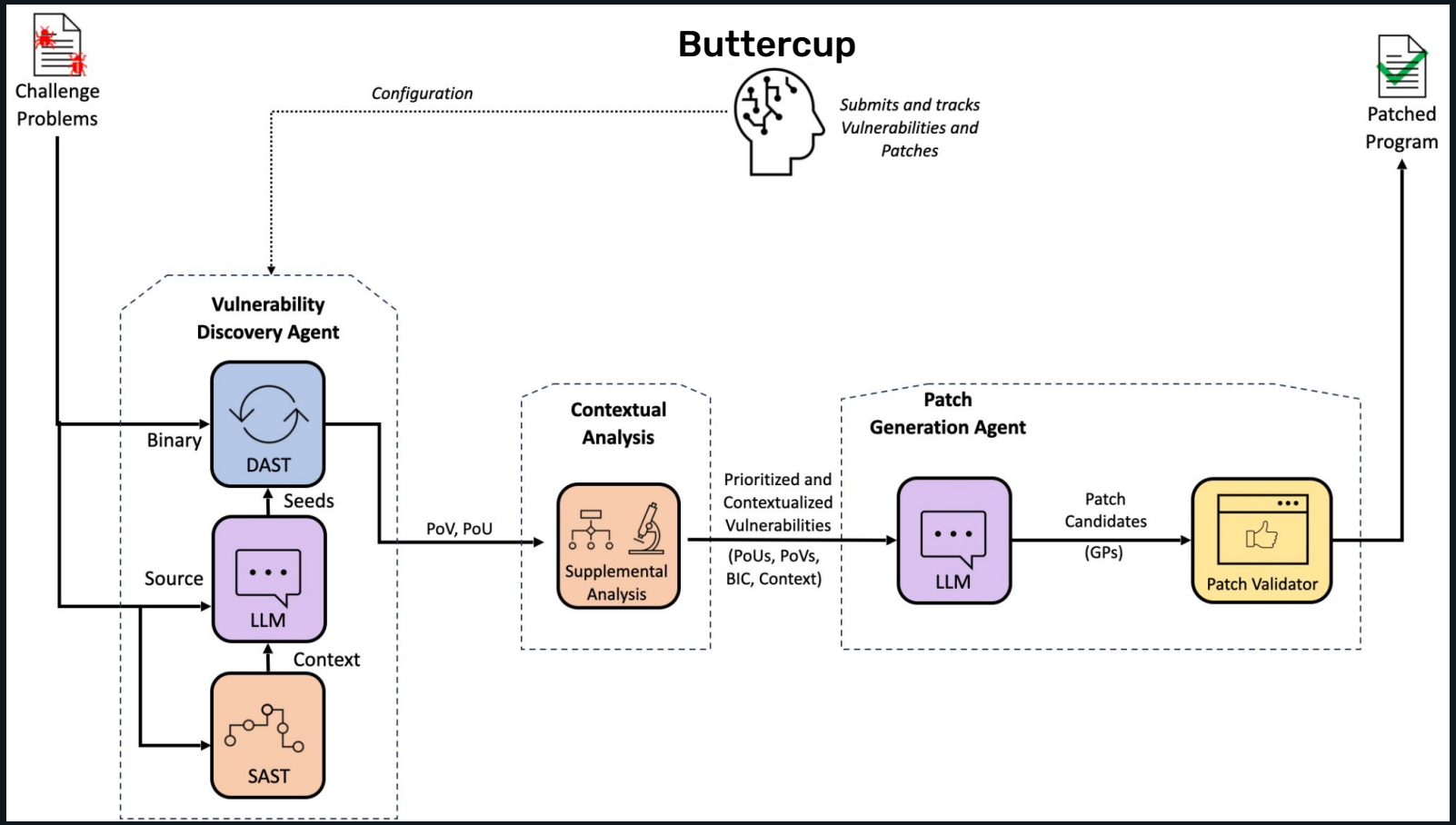
- Conventional software analysis works really well for certain problems.
- AI/ML-based analysis works really well for certain problems.
- Often, one approach works well where the other does not.

Break the problem down, use the best technique to solve each sub-problem.
Don’t expect LLMs to do things they aren’t good at!

Problem Breakdown

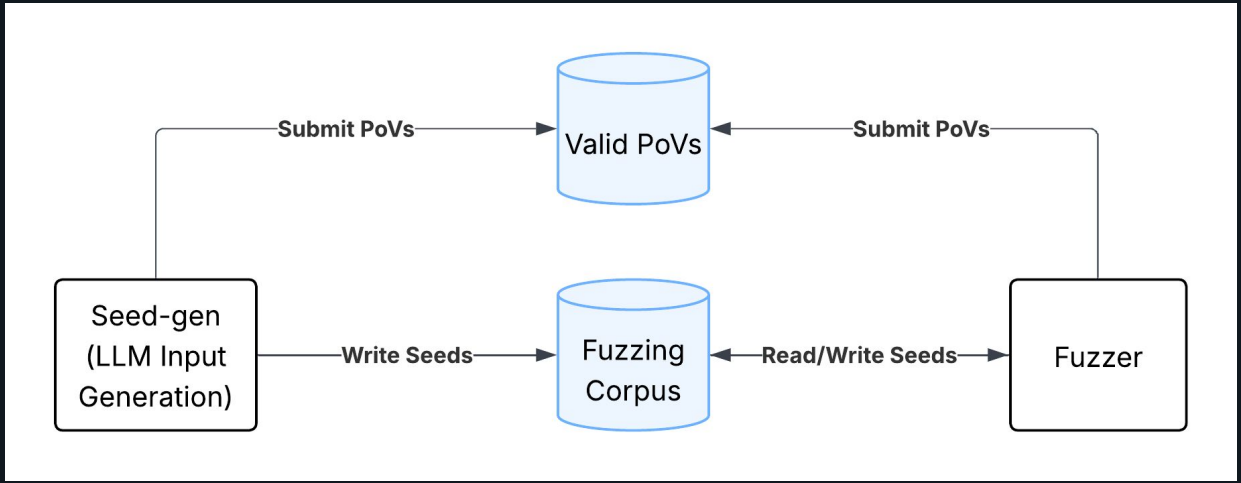
1. Discover / prove existence of vulnerabilities
2. Contextualize vulnerabilities
3. Create and validate patches
4. Orchestrate these tasks to:
    a. Effectively allocate resources
    b. Maximize score

Buttercup System Design



Vulnerability Discovery

Approach: Combine fuzzing and LLM input generation



Note: PoVs stands for Proofs of Vulnerability

Fuzzing

- **Standard OSS-Fuzz fuzzers:**
  - LibFuzzer for C/C++
  - Jazzer for Java
- **Fuzzer bots sample active harnesses to run short fuzz campaigns**
- **Merger bots save inputs which improve coverage**
  - Merge a fuzzer bot's local corpus to the shared corpus

Seed-gen

Design

- Several tasks that use LLMs to create seeds and/or PoVs
- All tasks use tools to collect context from the codebase before generating inputs

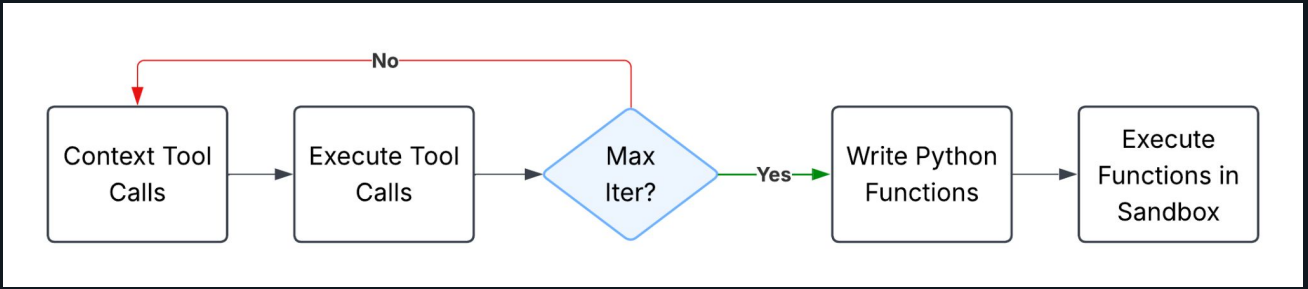
Goal 1: Support Fuzzing

- **Init task**: Bootstrap fuzzer with initial seed inputs that exercise harness
- **Explore task**: Increase coverage for a target function

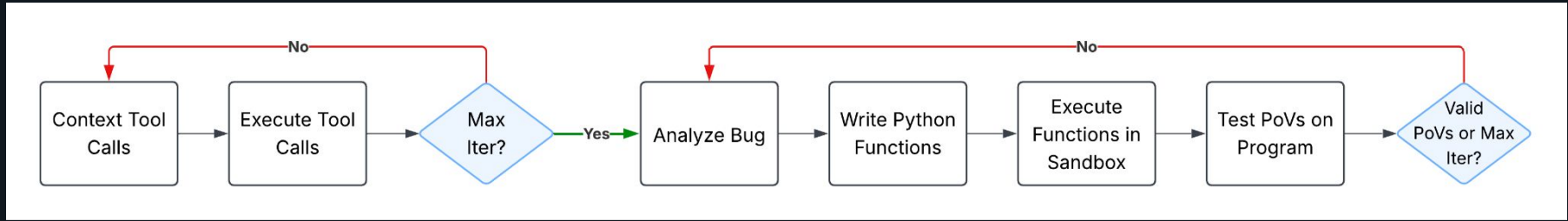
Goal 2: Independently Find Bugs

- **Vuln discovery task**: Identify and validate vulnerabilities in target to create PoVs
  - Most expensive task to thoroughly explore code and test hypotheses

Seed-gen Tasks



Initialization and Explore Tasks

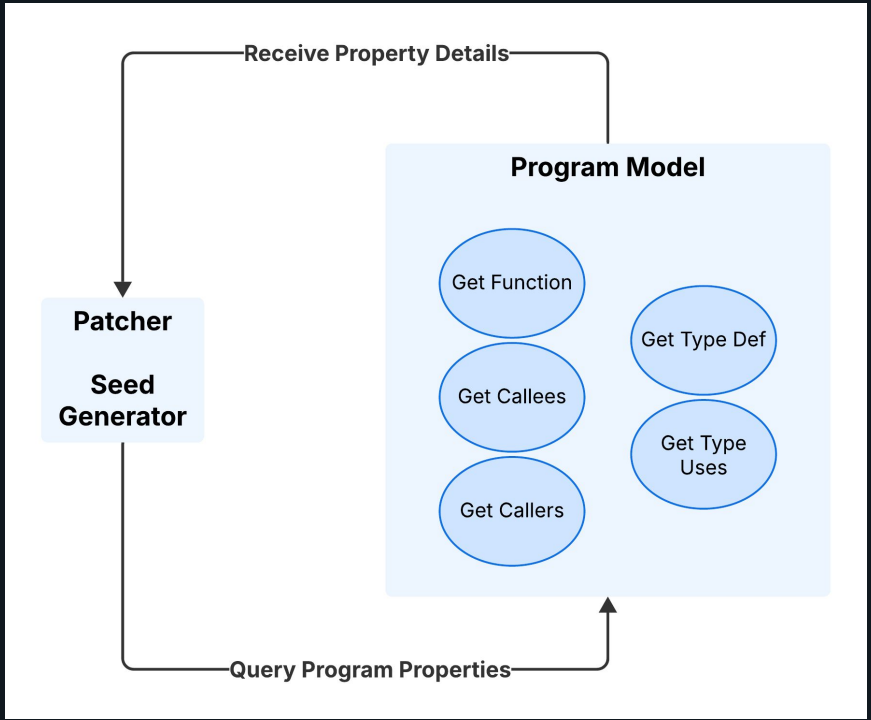


Vulnerability Discovery Task

Seed-gen Details

- **Model: Claude 4 Sonnet**
- **Uses Langchain/Langgraph**
- **Typical cost of task is cheap: <$1 to $5 (for vuln discovery)**
    - By design to prioritize breadth and resampling

Contextualization



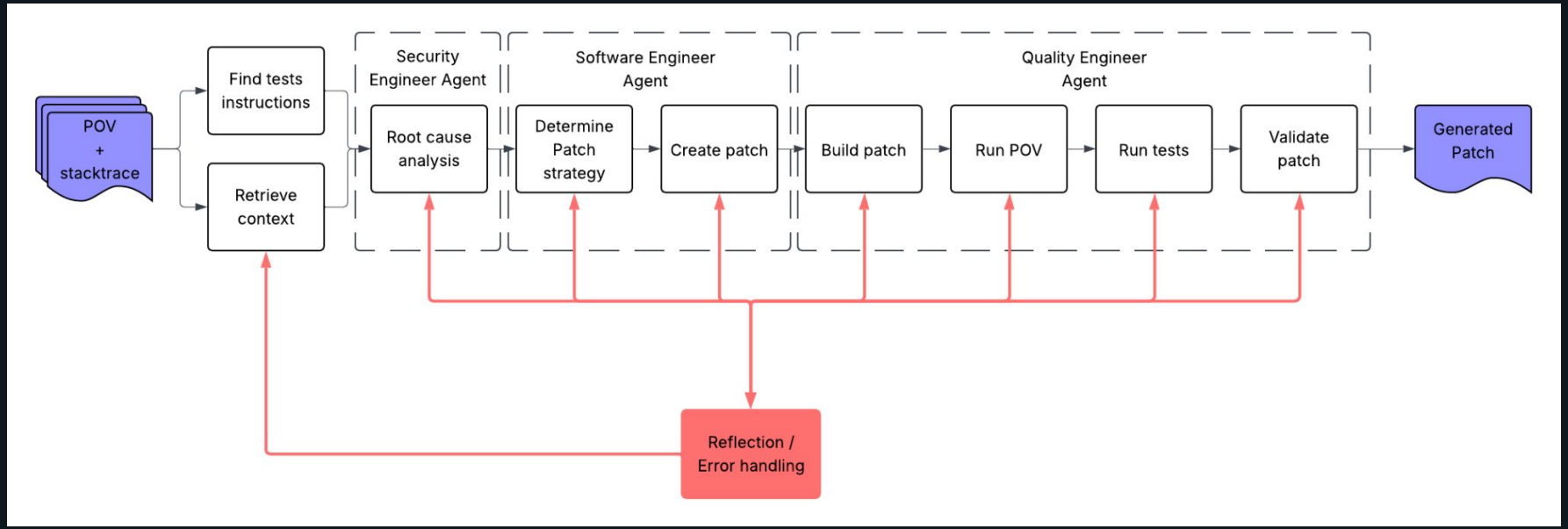
- Constructs program model using CodeQuery + Tree-sitter
- Supports querying program properties (functions & types)
- Called by LLMs from **Seed Generator** and **Patcher** as tools

Patcher

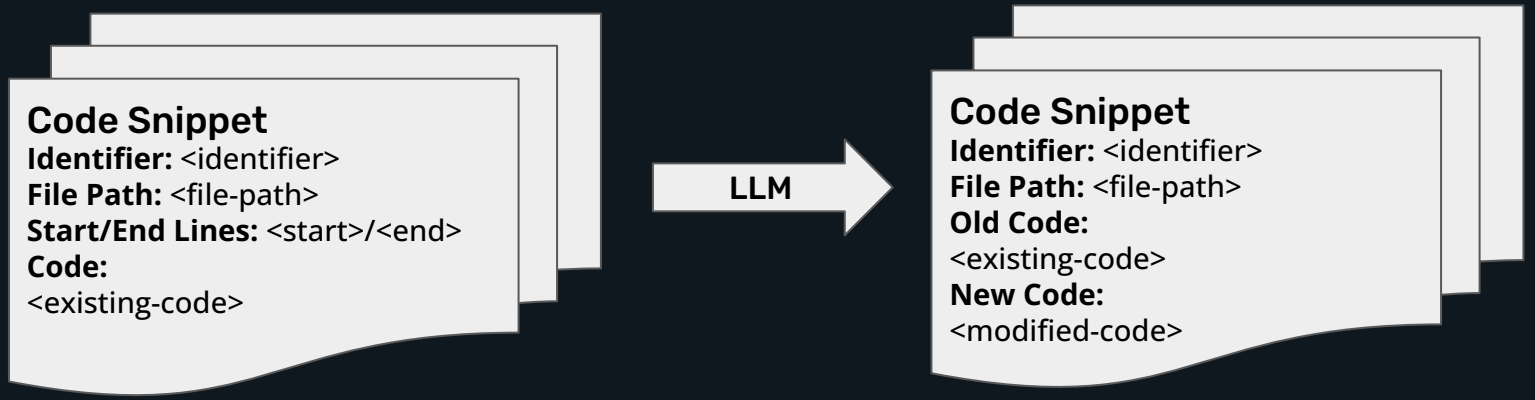
- LLM-based multi-agent system
  - Software, Security, and Quality Engineer Agents working together
- Programmatic agents hand-off
  - Data flow between agent is (mostly) deterministic
  - More control over the process
  - Error handling relies on LLMs to determine resolution steps

- **Implementation**
  - Less than 6K LOC, Python
  - LangChain/LangGraph
  - Preferred model: OpenAI/GPT-4.1

Patcher Flow



Patch Creation



Buttercup After AIxCC

- **Updated Buttercup to easily run on a standard laptop**
- **C++ Support**
- **Gemini models**
- **We have plans to keep improving Buttercup!**
  - Support more languages
  - Agentic workflows
  - More tools for agents

How to use Buttercup?

Requires:

- Machine with 8 CPU, 16 GB RAM
- LLM API Key (OpenAI, Anthropic, or Gemini)
- OSS-Fuzz compatible project as a target

Easy to use:

- Seamless script to configure Buttercup
- Web UI to task Buttercup and monitor results

Repo: github.com/trailofbits/buttercup

Learn More

Try out Buttercup:
github.com/trailofbits/buttercup

Website: trailofbits.com

Careers: trailofbits.com/careers

Blog: blog.trailofbits.com

Buttercup