Buckle Up, Buttercup: Our Experience Competing in the AI Cyber Challenge

9 August 2025

Our Team

Michael D. Brown

Overall Team Lead
Lead Designer of Buttercup

Ian Smith

Vuln Discovery Lead
Co-Designer of Buttercup

Ronald Eytchison

AI-Based Seed Generation Lead

Our Team

Henrik Brodin

Orchestration Lead
(Finals)

Eric Kilmer

Orchestration Co-Lead
(Semi-Finals)

Francesco Bertolaccini

Orchestration Co-Lead
(Semi-Finals)

Our Team

Riccardo Schirrone

Patcher Lead

Evan Downing

Contextualization Lead

Boyan Milanov

System Developer

Our Team

Alessandro Gario

Challenge Creator
(Internal Red Team)

Brad Swain

Challenge Creator
(Internal Red Team)

Our Team

Akshay Kumar

Challenge Creation
(Semi-finals)

Will Tan

Systems Developer
(Semi-finals)

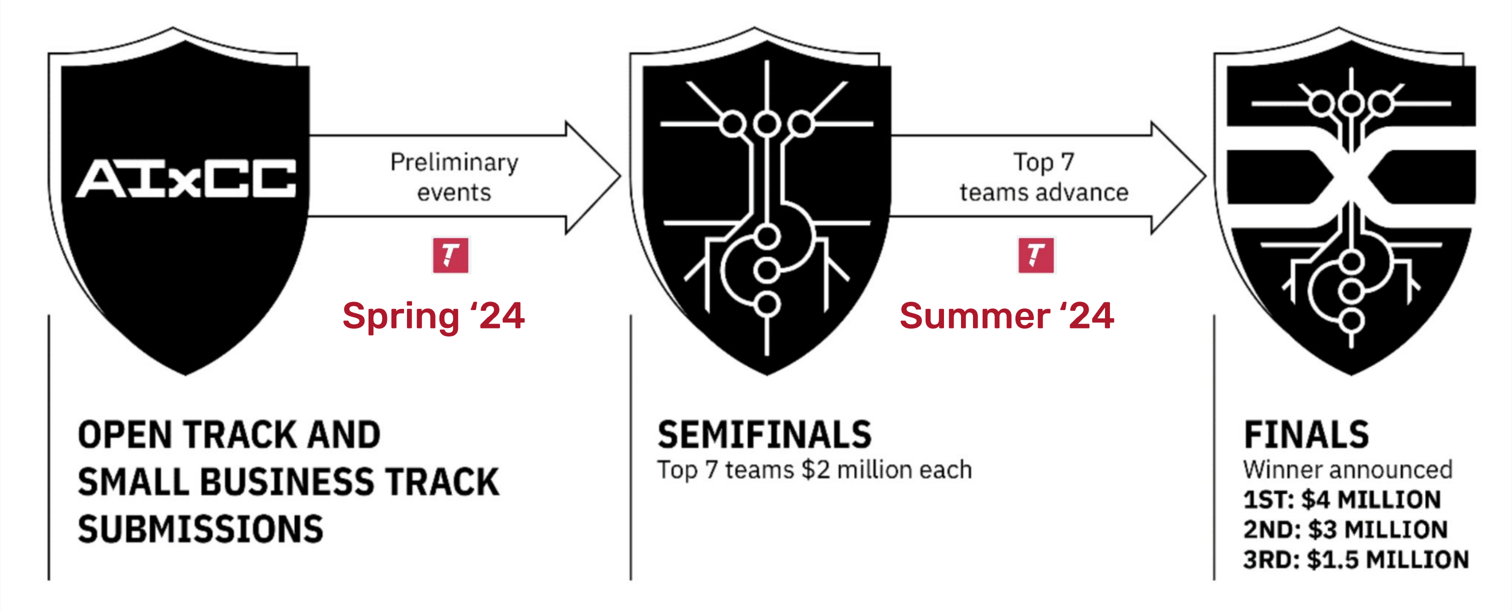
Alan Cao

Systems Developer
(Semi-finals)

A Brief Origin Story

AI Cyber Challenge (AIxCC)

AIxCC is a competition to design a novel automated AI system (CRS) that can find and patch bugs in real-world open-source software.



Spring '24

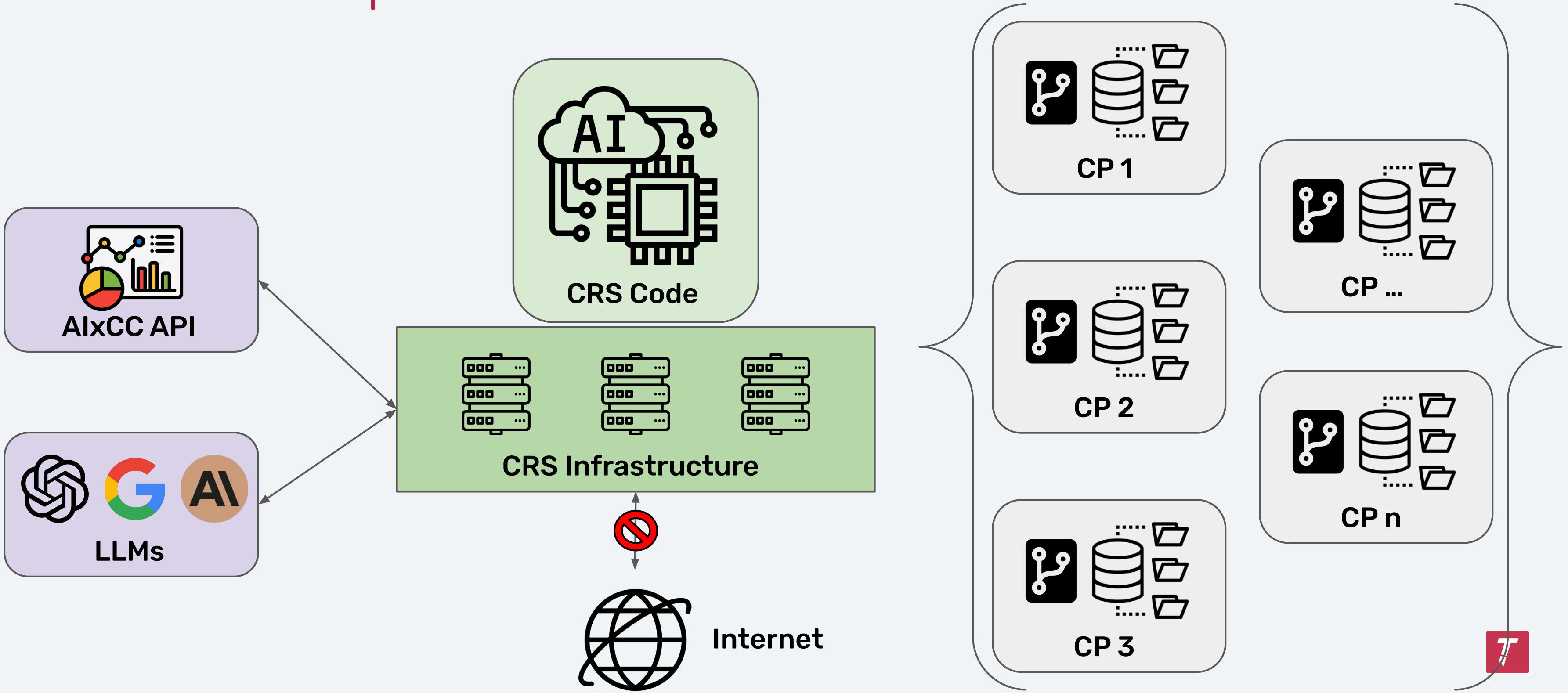
Summer '24

OPEN TRACK AND SMALL BUSINESS TRACK SUBMISSIONS

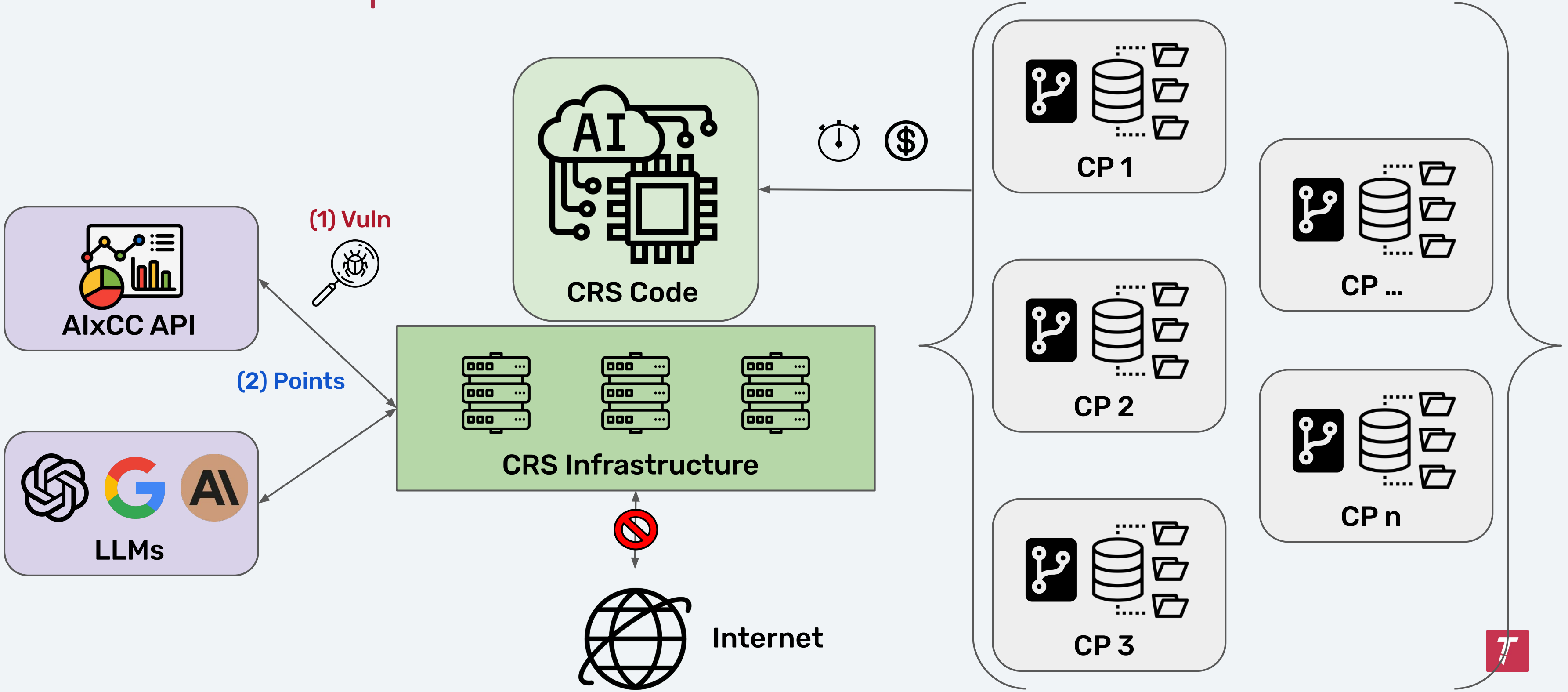
SEMIFINALS
Top 7 teams $2 million each

FINALS
Winner announced
1ST: $4 MILLION
2ND: $3 MILLION
3RD: $1.5 MILLION

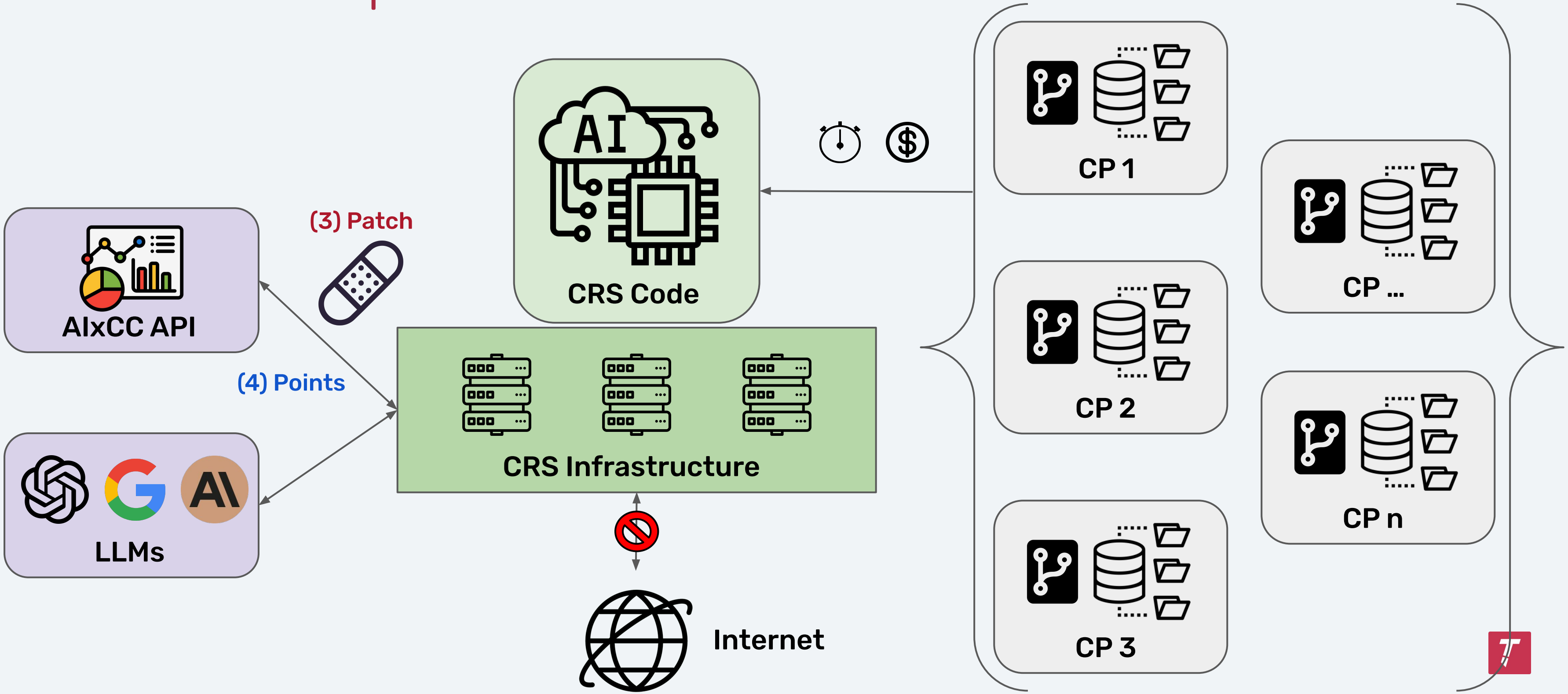
AIxCC Competition Structure



AIxCC Competition Structure



AIxCC Competition Structure



Buttercup’s Design

Our Approach

Guiding Principles

- 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.

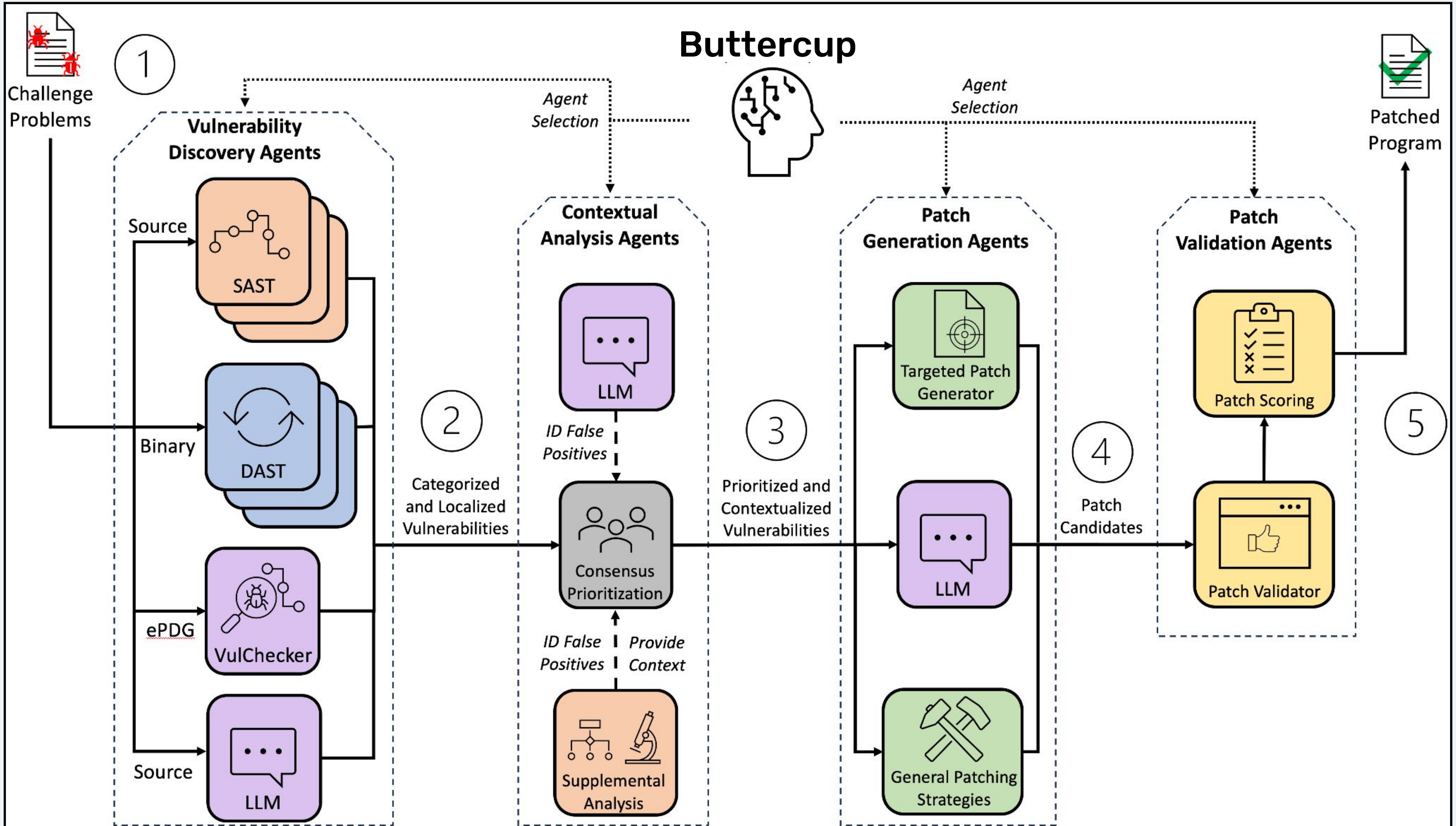
Break the problem down, use the best technique to solve each sub-problem.

Don’t expect LLMs to do things they aren’t good at!

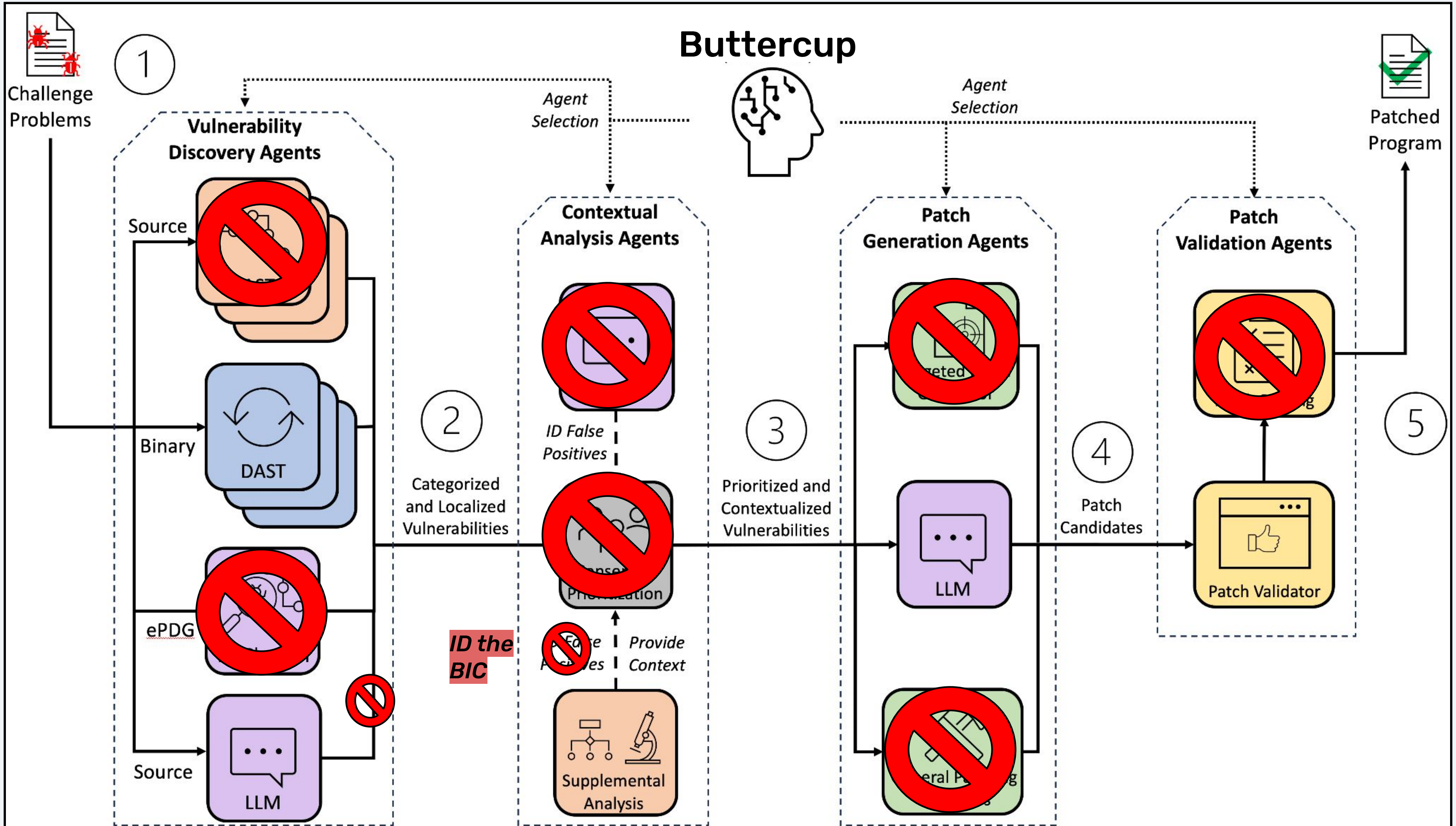
Problem Breakdown

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

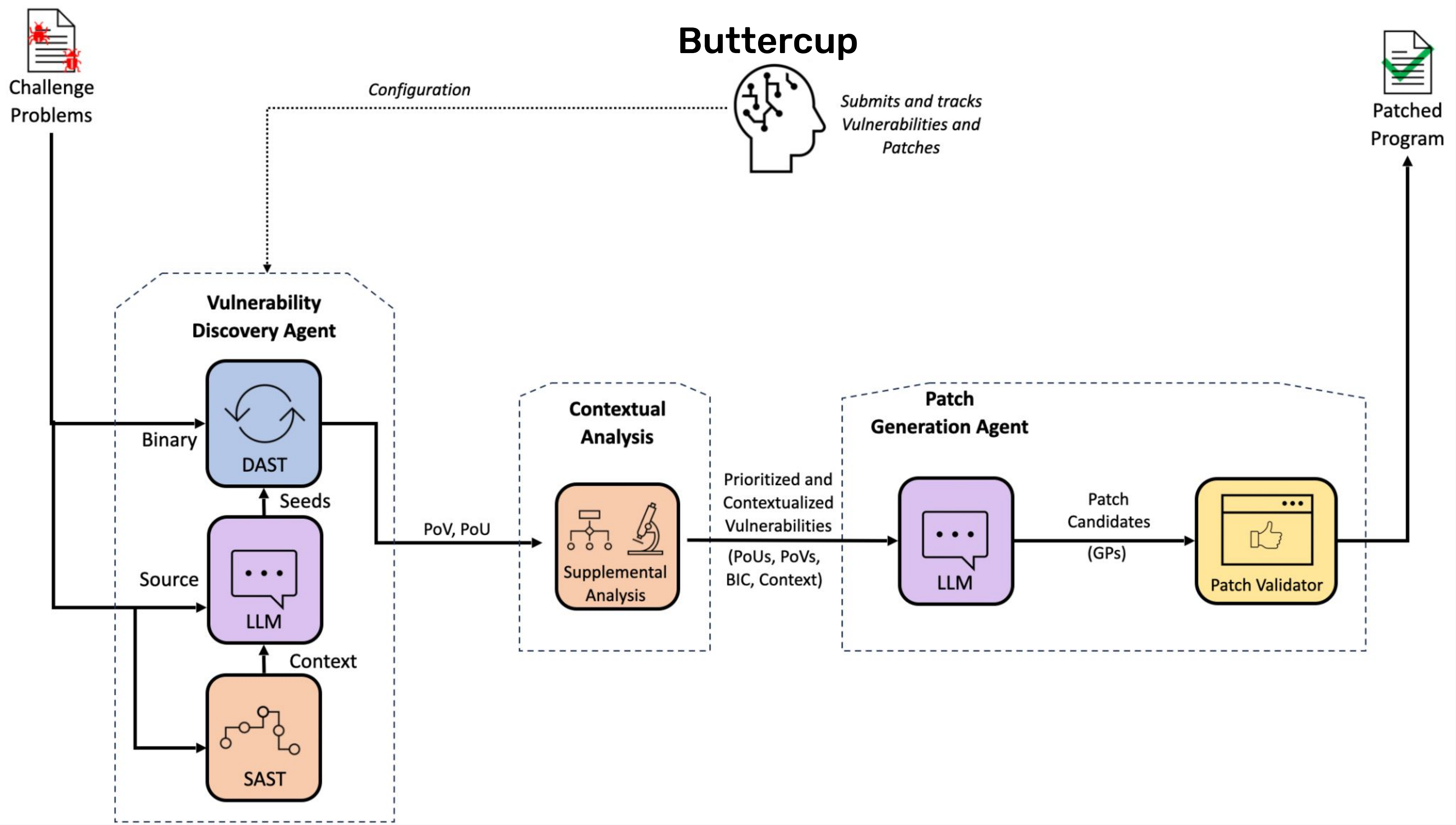
CRS Architecture (Concept Paper)



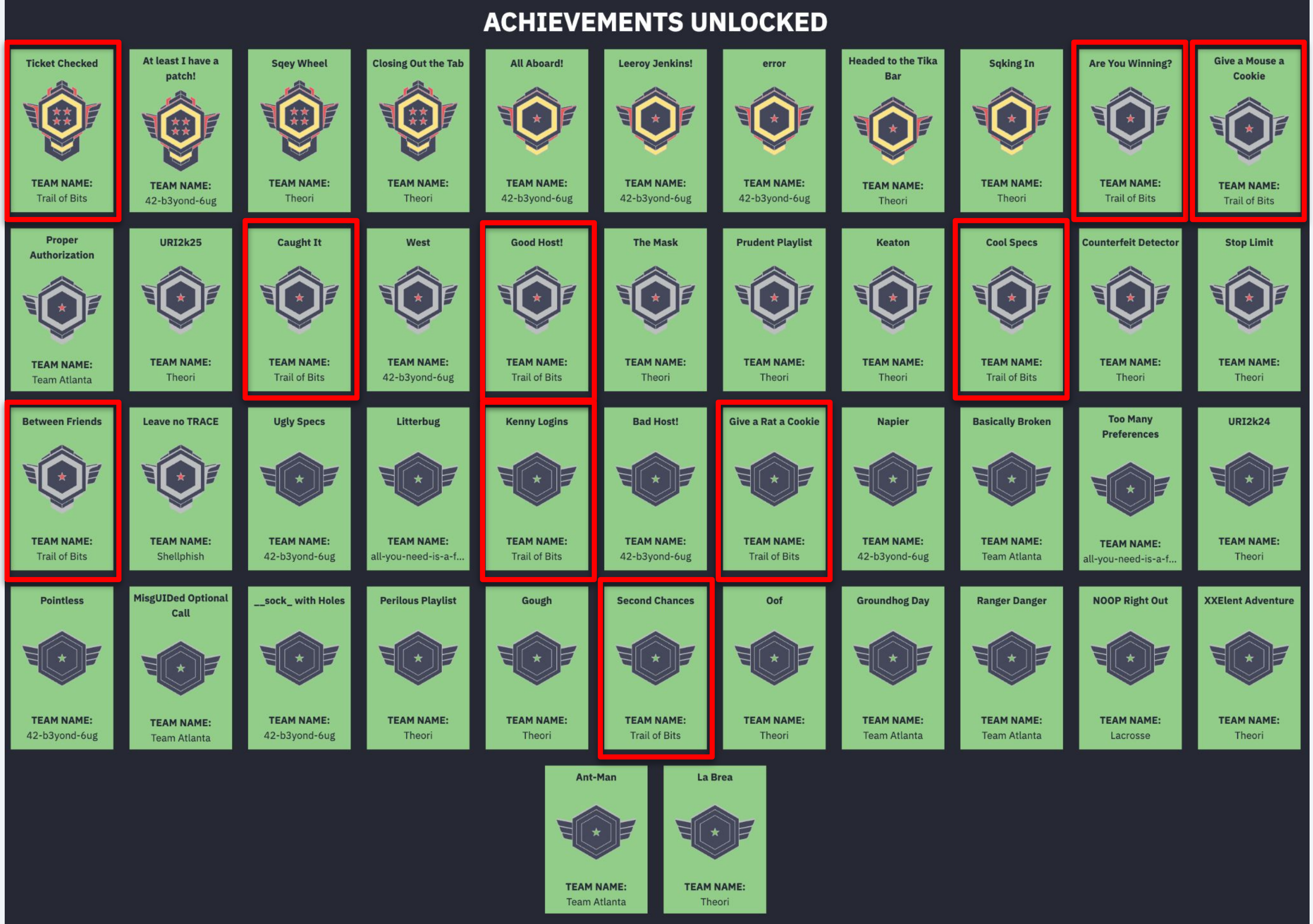
CRS Architecture



CRS Architecture (Competition)



Buttercup in the Semifinals



ACHIEVEMENTS UNLOCKED

Ticket Checked
TEAM NAME:
Trail of Bits

At least I have a patch!
TEAM NAME:
42-b3yond-6ug

Sqey Wheel
TEAM NAME:
Theori

Closing Out the Tab
TEAM NAME:
Theori

All Aboard!
TEAM NAME:
42-b3yond-6ug

Leeroy Jenkins!
TEAM NAME:
42-b3yond-6ug

error
TEAM NAME:
42-b3yond-6ug

Headed to the Tika Bar
TEAM NAME:
Theori

Sqking In
TEAM NAME:
Theori

Are You Winning?
TEAM NAME:
Trail of Bits

Give a Mouse a Cookie
TEAM NAME:
Trail of Bits

Proper Authorization
TEAM NAME:
Team Atlanta

URI2k25
TEAM NAME:
Theori

Caught It
TEAM NAME:
Trail of Bits

West
TEAM NAME:
42-b3yond-6ug

Good Host!
TEAM NAME:
Trail of Bits

The Mask
TEAM NAME:
Theori

Prudent Playlist
TEAM NAME:
Theori

Keaton
TEAM NAME:
Theori

Cool Specs
TEAM NAME:
Trail of Bits

Counterfeit Detector
TEAM NAME:
Theori

Stop Limit
TEAM NAME:
Theori

Between Friends
TEAM NAME:
Trail of Bits

Leave no TRACE
TEAM NAME:
Shellphish

Ugly Specs
TEAM NAME:
42-b3yond-6ug

Litterbug
TEAM NAME:
all-you-need-is-a-f...

Kenny Logins
TEAM NAME:
Trail of Bits

Bad Host!
TEAM NAME:
42-b3yond-6ug

Give a Rat a Cookie
TEAM NAME:
Trail of Bits

Napier
TEAM NAME:
42-b3yond-6ug

Basically Broken
TEAM NAME:
Team Atlanta

Too Many Preferences
TEAM NAME:
all-you-need-is-a-f...

URI2k24
TEAM NAME:
Theori

Pointless
TEAM NAME:
42-b3yond-6ug

MisgUIDed Optional Call
TEAM NAME:
Team Atlanta

__sock_ with Holes
TEAM NAME:
42-b3yond-6ug

Perilous Playlist
TEAM NAME:
Theori

Gough
TEAM NAME:
Theori

Second Chances
TEAM NAME:
Trail of Bits

Oof
TEAM NAME:
Theori

Groundhog Day
TEAM NAME:
Team Atlanta

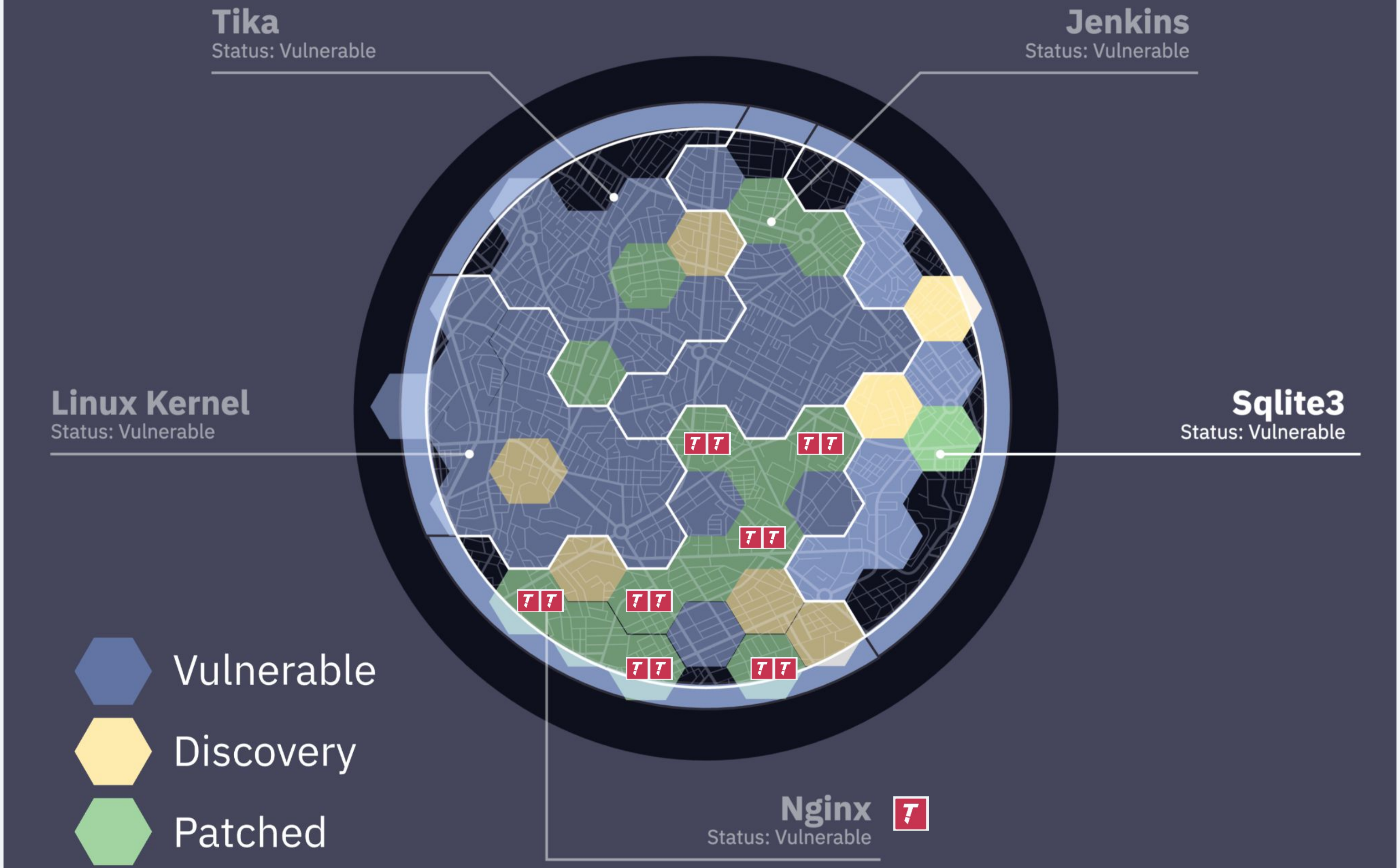
Ranger Danger
TEAM NAME:
Team Atlanta

NOOP Right Out
TEAM NAME:
Lacrosse

XXElent Adventure
TEAM NAME:
Theori

Ant-Man
TEAM NAME:
Team Atlanta

La Brea
TEAM NAME:
Theori



Tika
Status: Vulnerable

Jenkins
Status: Vulnerable

Linux Kernel
Status: Vulnerable

Sqlite3
Status: Vulnerable

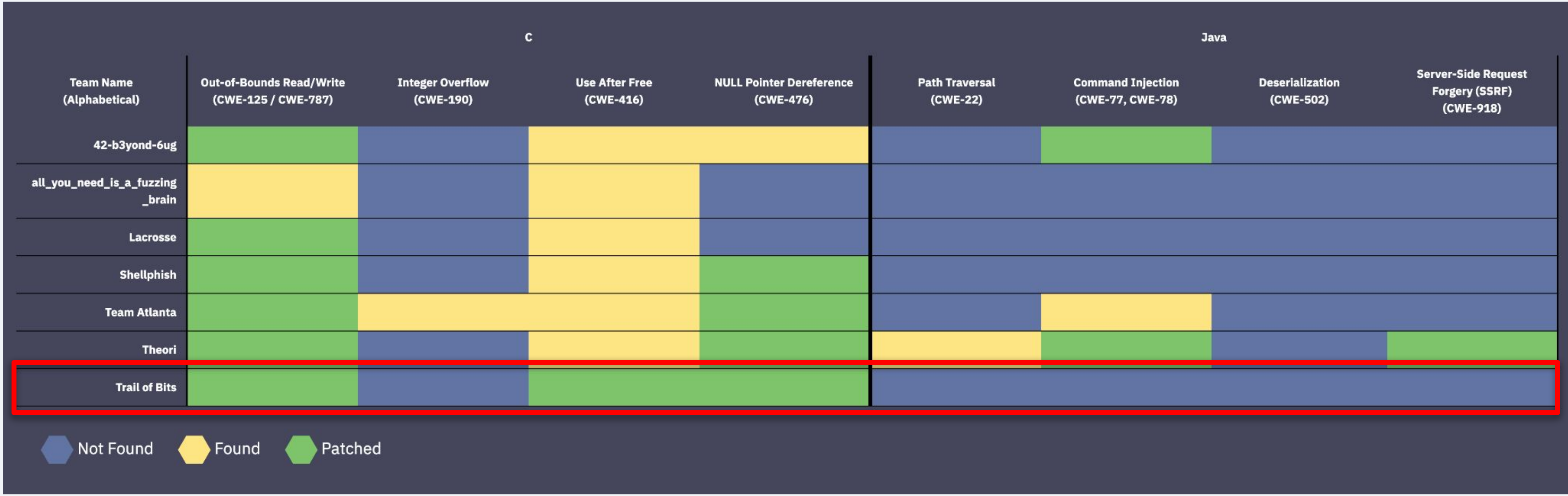
Nginx
Status: Vulnerable

Vulnerable

Discovery

Patched

Performance by CWE type



	C				Java			
Team Name (Alphabetical)	Out-of-Bounds Read/Write (CWE-125 / CWE-787)	Integer Overflow (CWE-190)	Use After Free (CWE-416)	NULL Pointer Dereference (CWE-476)	Path Traversal (CWE-22)	Command Injection (CWE-77, CWE-78)	Deserialization (CWE-502)	Server-Side Request Forgery (SSRF) (CWE-918)
42-b3yond-6ug	🟩	🟦	🟨	🟨	🟦	🟩	🟦	🟦
all_you_need_is_a_fuzzing_brain	🟨	🟦	🟨	🟦	🟦	🟦	🟦	🟦
Lacrosse	🟩	🟦	🟨	🟦	🟦	🟦	🟦	🟦
Shellphish	🟩	🟦	🟨	🟩	🟦	🟦	🟦	🟦
Team Atlanta	🟩	🟨	🟨	🟩	🟦	🟨	🟦	🟦
Theori	🟩	🟦	🟨	🟩	🟨	🟩	🟦	🟩
Trail of Bits	🟩	🟦	🟩	🟩	🟦	🟦	🟦	🟦

🟦 Not Found 🟨 Found 🟩 Patched

Buttercup 2.0

How did Buttercup evolve for the finals?

Lessons Learned from semi-finals:

- Validated our overall approach
- Need better testing / handling of Java challenges
- CWE-type specific seed-generation may have helped

Rule changes for finals:

- Massive scale and budget (time, compute, and AI) increases
- Several exhibition rounds
- More complex scoring (SARIFs, bundles, duplication penalties)
- Custom AI/ML models allowed

Building Buttercup 2.0

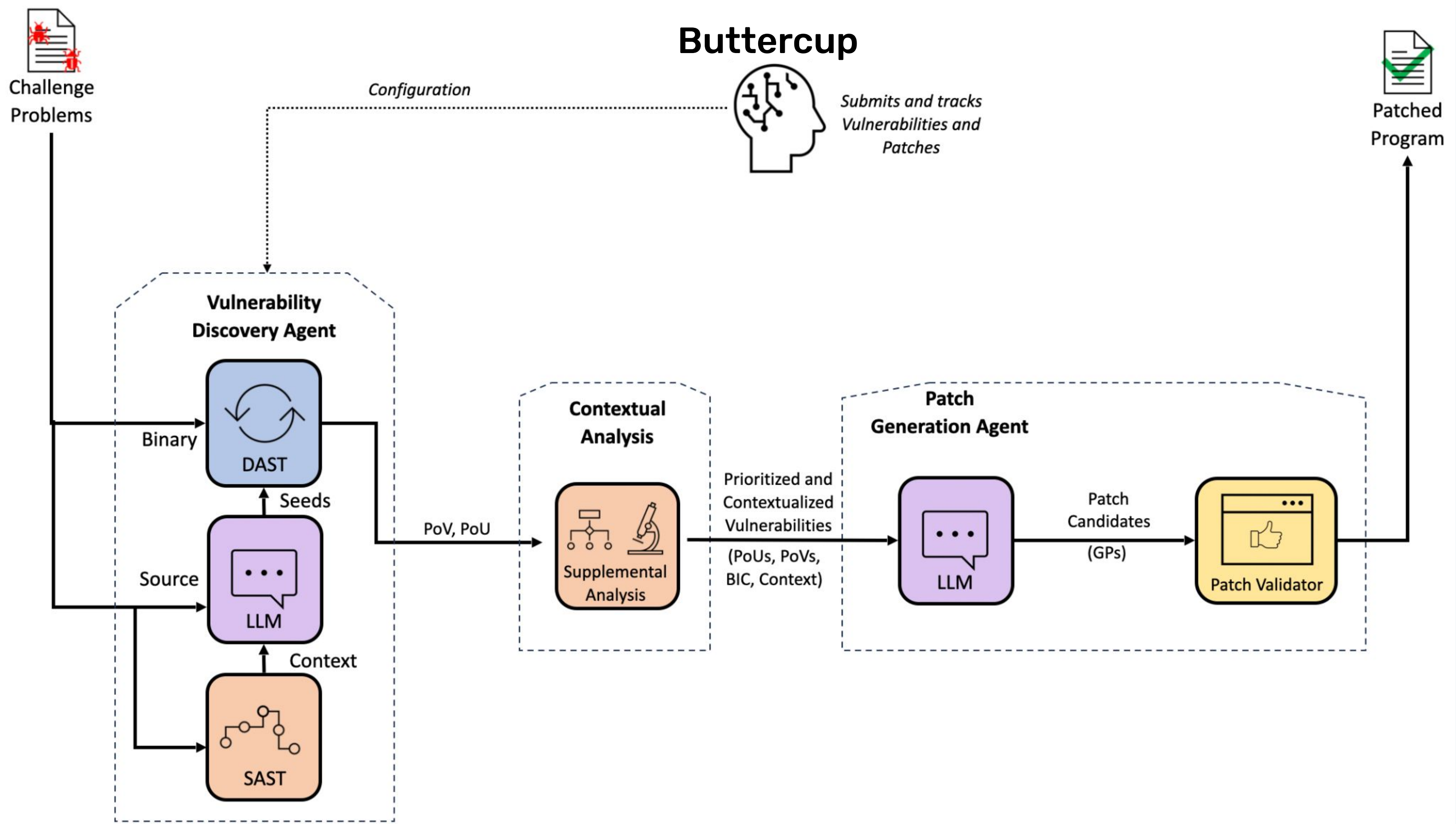
Buttercup 2.0 is essentially a from-scratch rebuild.

Driven by need for:

- more technically complex analysis components
- ability to easily change scale / cost of deployment for various rounds
- high degree of reliability and robustness to errors

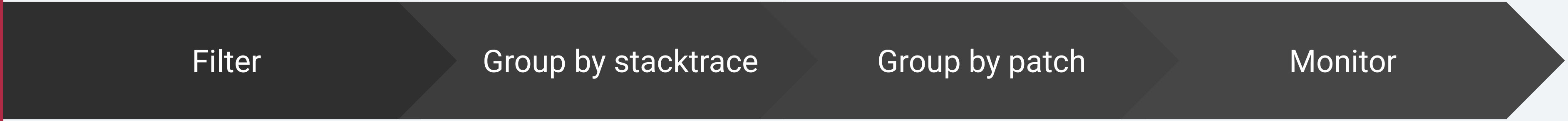
Still, our high-level Buttercup remained the same as the semi-finals

CRS Architecture (Competition)



Buttercup 2.0 Technical Details

Orchestration - Submission Processing



Filter

Group by stacktrace

Group by patch

Monitor

Vulnerability discovery produces many PoVs - filter stack traces already seen

Group PoVs with similar stack traces - examples of the same underlying vulnerability.

Group PoVs remediated by the same patch - same underlying vulnerability

As new PoVs come in merge by fuzzy stack match and patches. Rebuild bundles as needed.

PoV - Proof of Vulnerability

Vulnerability Discovery

- **Strategy: Combine fuzzing and LLM input generation**
- Use standard OSS-Fuzz fuzzers:
  - LibFuzzer for C/C++
  - Jazzer for Java
- Fuzzer bots sample active harnesses to run short fuzz campaigns
- Fuzzing corpus:
  - Merger bots merge a fuzzer bot’s local corpus to the shared corpus
  - LLM input generation also submits to the corpus

Vulnerability Discovery: LLM “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

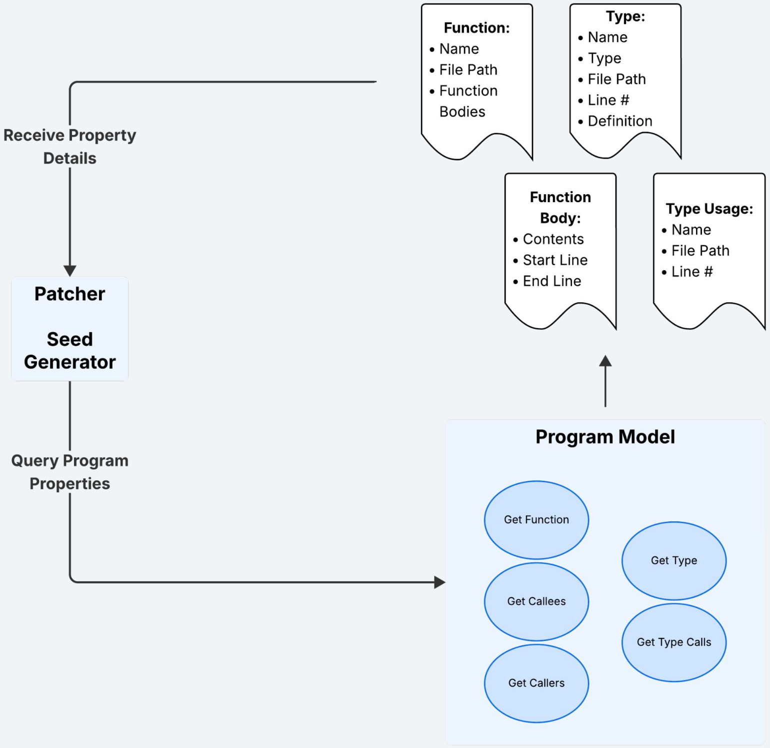
Goal 1: Support Fuzzing

- **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

Contextualization



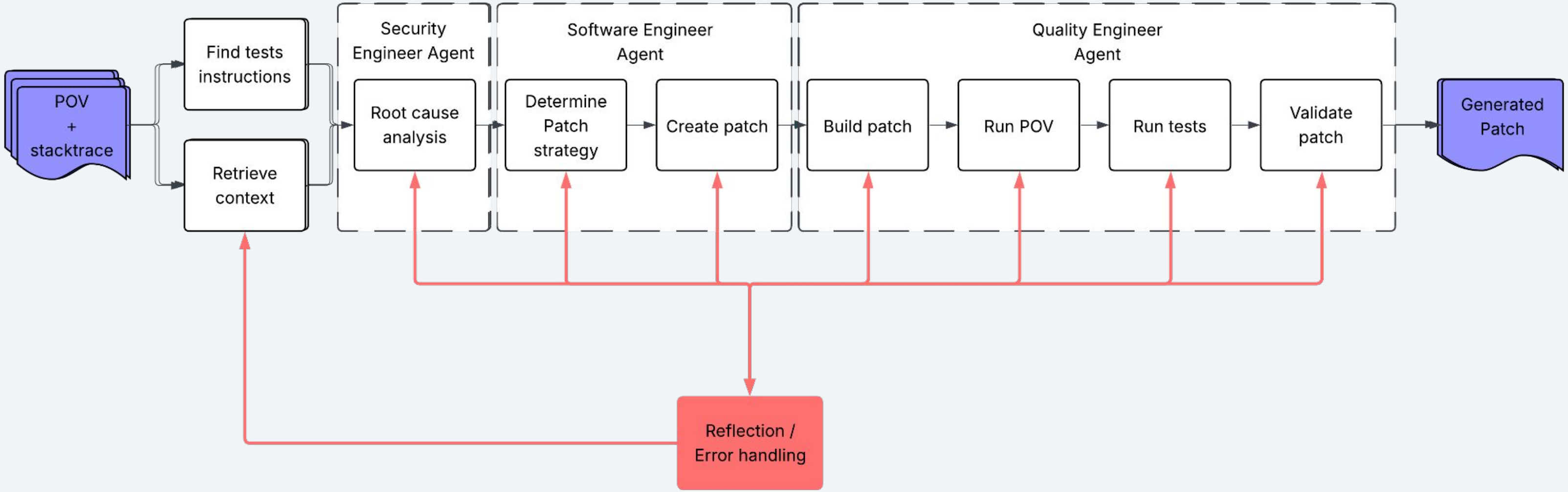
- Constructs program model using CodeQuery + Tree-sitter
- Supports querying program properties (functions & types)
- Called by LLMs from **Seed Generator** and **Patcher** using LangGraph's Tool library

Patcher

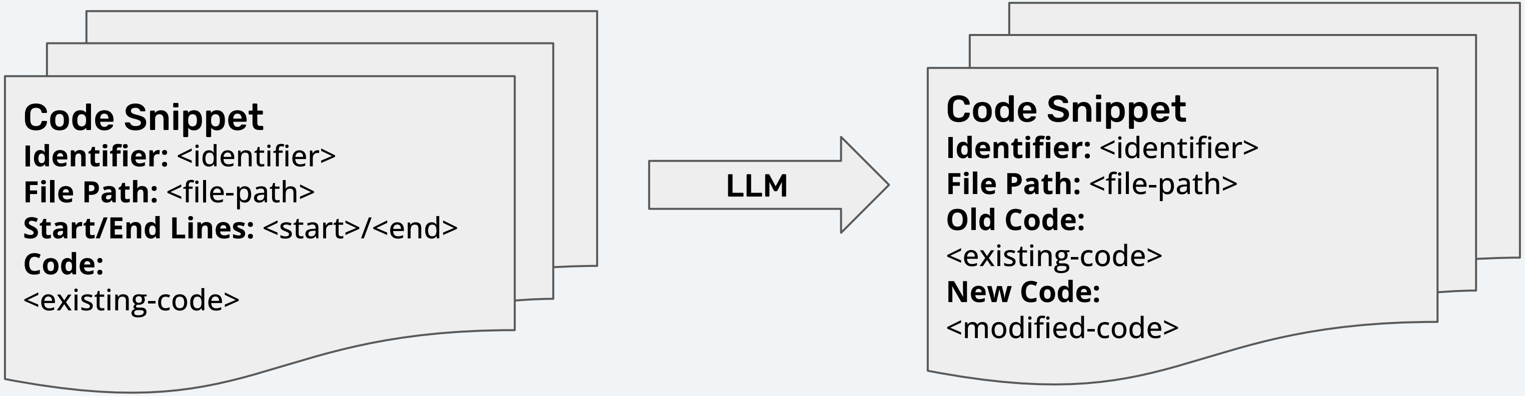
- LLM-based multi-agent system
  - Software, Security, and Quality Engineer Agents working together
- Programmatic agents hand-off
  - Data flow between agent is (mostly) deterministic
  - More control over the process
  - Error handling relies on LLMs to determine resolution steps

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

Patcher: flow



Patcher: patch creation



Buttercup in the Finals

How did Buttercup do in Exhibition Rounds?

Buttercup was the best performing CRS in Round 1:

- Found and patched a vulnerability in both challenges with 100% accuracy
- Used only ~$1000 of available $30,000 budget

But we crashed hard in Round 2:

- Issue with filename length in vulnerability discovery component
- Caused a hard failure after only 3/18 challenges were processed
- We later reproduced Round 2 and Buttercup was successful on all challenges

And bounced back in Round 3:

- Buttercup found and/or patched vulnerabilities in 20/26 challenges!

How did Buttercup do in the scored round?

Buttercup came in second place, winning $3 million!

- Found 28 vulnerabilities, patched 19
- Used only ~$40,000 of available budget
- ~90% Accuracy
- Found at least one PoV no one else did
- Found at least one non-synthetic vulnerability

Keys to success:

- Accuracy
- Scoring well across all tasks

I want to try Buttercup!

You’re in Luck….

Buttercup is Open Source!

The exact code we submitted for the semi-finals and finals code is available on our company github organization!

- Buttercup 1.0 https://github.com/trailofbits/asc-buttercup
- Buttercup 2.0 https://github.com/trailofbits/afc-buttercup

Fair warning: Buttercup was designed to run on competition infrastructure and at massive scale, so this version of Buttercup isn’t terribly user friendly…

And we'll do you one better!

A standalone variant of Buttercup is also available!

We've also created a version of Buttercup that runs on commodity (laptop) and typical server-grade hardware. You can check it out at:

- Buttercup standalone https://github.com/trailofbits/buttercup

Enjoy!

Thanks for Coming!