



# All You Need Is a Fuzzing Brain: A Retrospective

Jeff Huang, Ze Sheng, Qingxiao Xu, Matthew Woodcock

Texas A&M University

# Open Source

**Full CRS** (identical to what's used in the aixcc final round)

<https://github.com/o2lab/afc-crs-all-you-need-is-a-fuzzing-brain>

**Local version** (runs on a single VM w/ docker image):

- <https://github.com/o2lab/afc-crs-all-you-need-is-a-fuzzing-brain/tree/jeff>
- 

**Semi-final CRS:** <https://github.com/o2lab/asc-crs-all-you-need-is-a-fuzzing-brain>

All You Need Is a Fuzzing  
Brain

All You Need Is a Fuzzing  
Brain

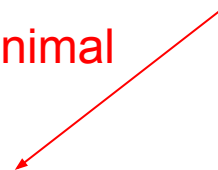
All You Need Is a Fuzzing  
Brain

LLMs



All You Need Is a Fuzzing

minimal



Brain

\*\*\*

\*\*\* NOTE: fuzzing was not performed, you have only  
\*\*\* executed the target code on a fixed set of inputs.

\*\*\*

# Agenda

- **What did we learn from the AIxCC competition?**
- **What surprised us and what challenged us the most?**
- **What motivated Flagship vs. Open Source models?**
- **What should AI cybersecurity pursue next?**



## What did we learn from the AlxCC competition?

- First of all, AlxCC is a great challenge!
- Today's LLMs are already super good at discovering and patching code vulnerabilities
- Building a **reliable** AI-based system is not easy (actually very hard!)

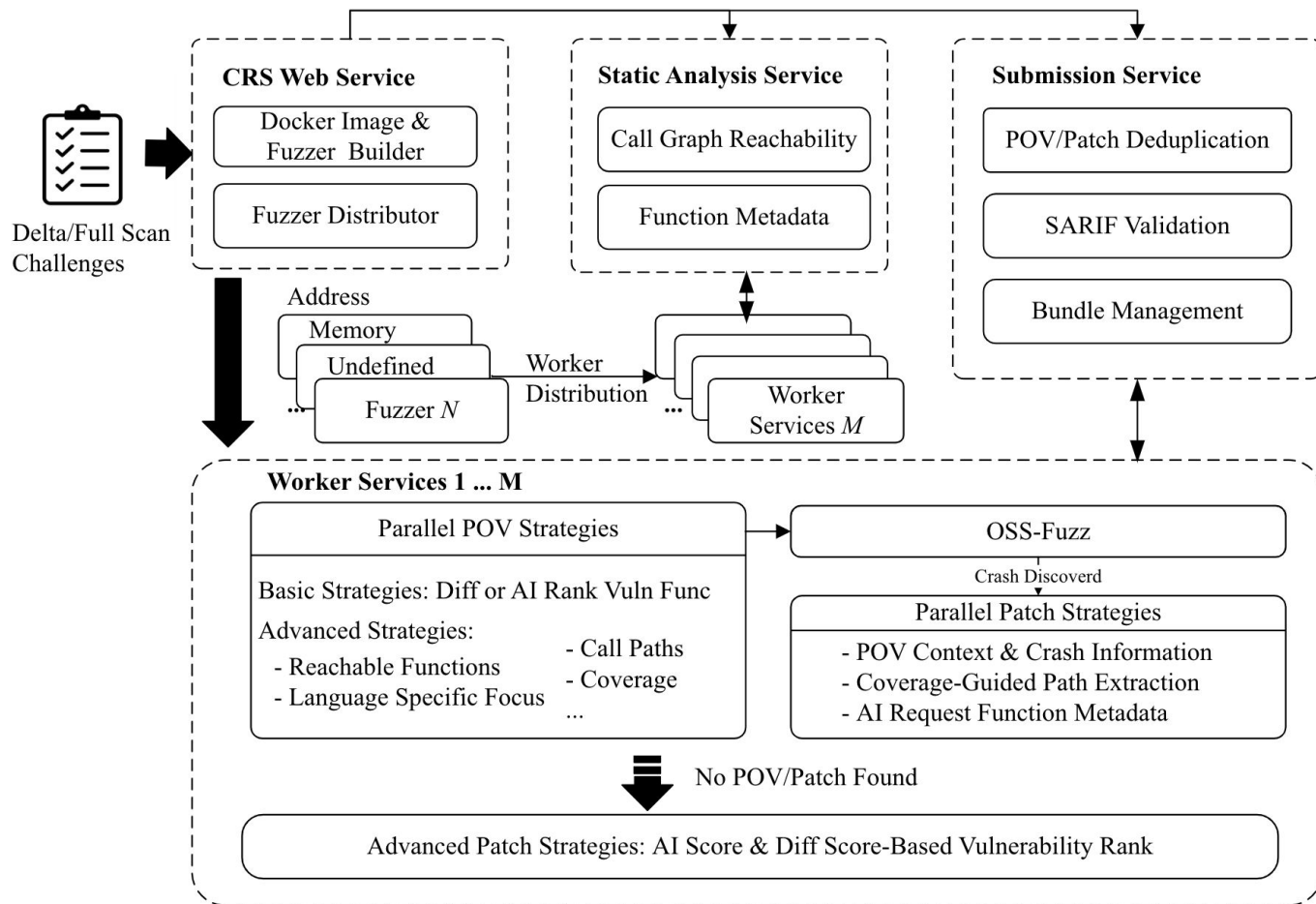
## Fastest POV & Fastest Patch

- Almost all our POV/Patch  within 20mins
- Some within *5 mins* from POV to Patch end-to-end

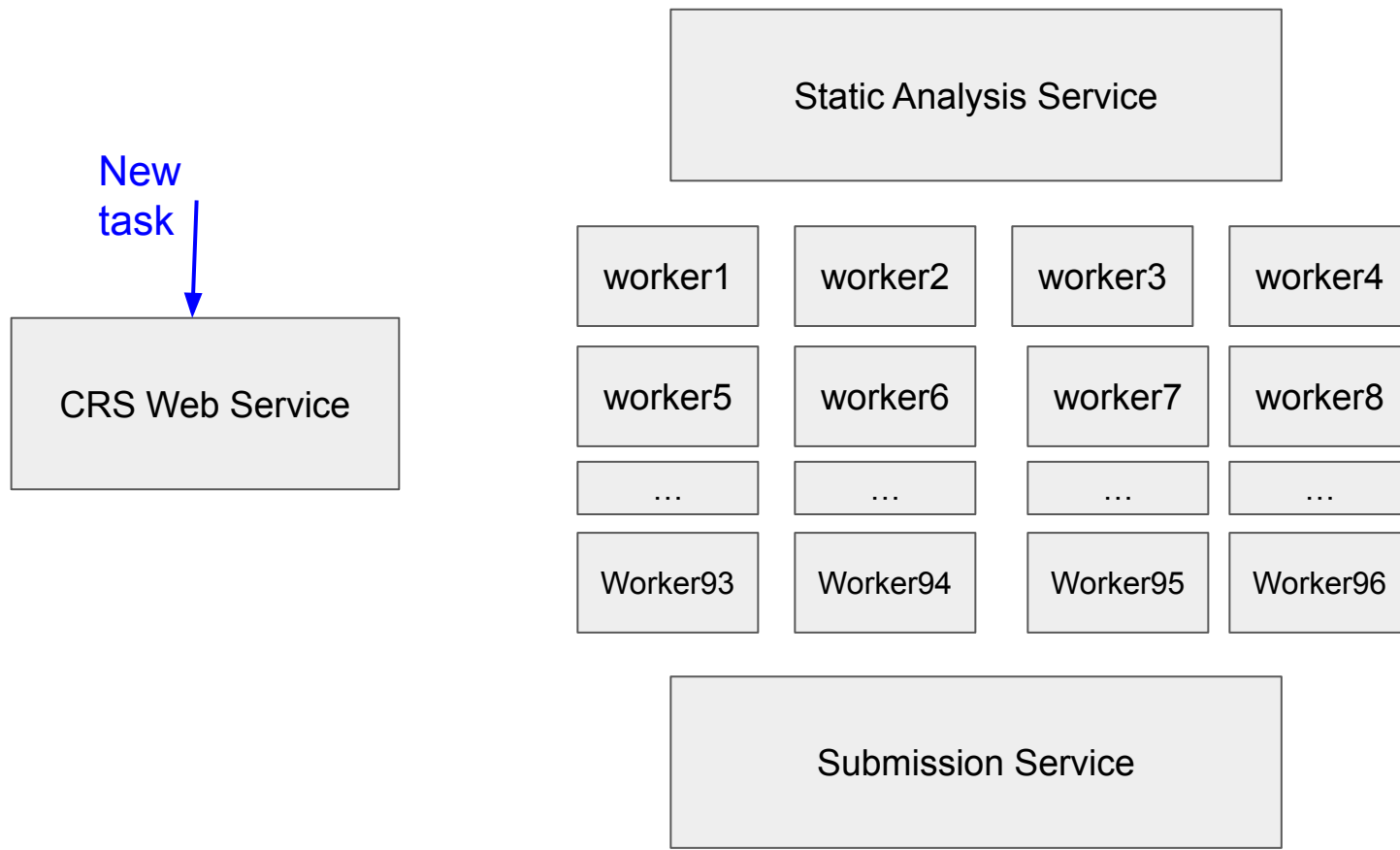
## Parallelization: *parallelize almost everything we can parallelize*

- *Allocated ~100 VM (each 32-192 cores)*
- *Each VM runs 100-10K threads simultaneously*

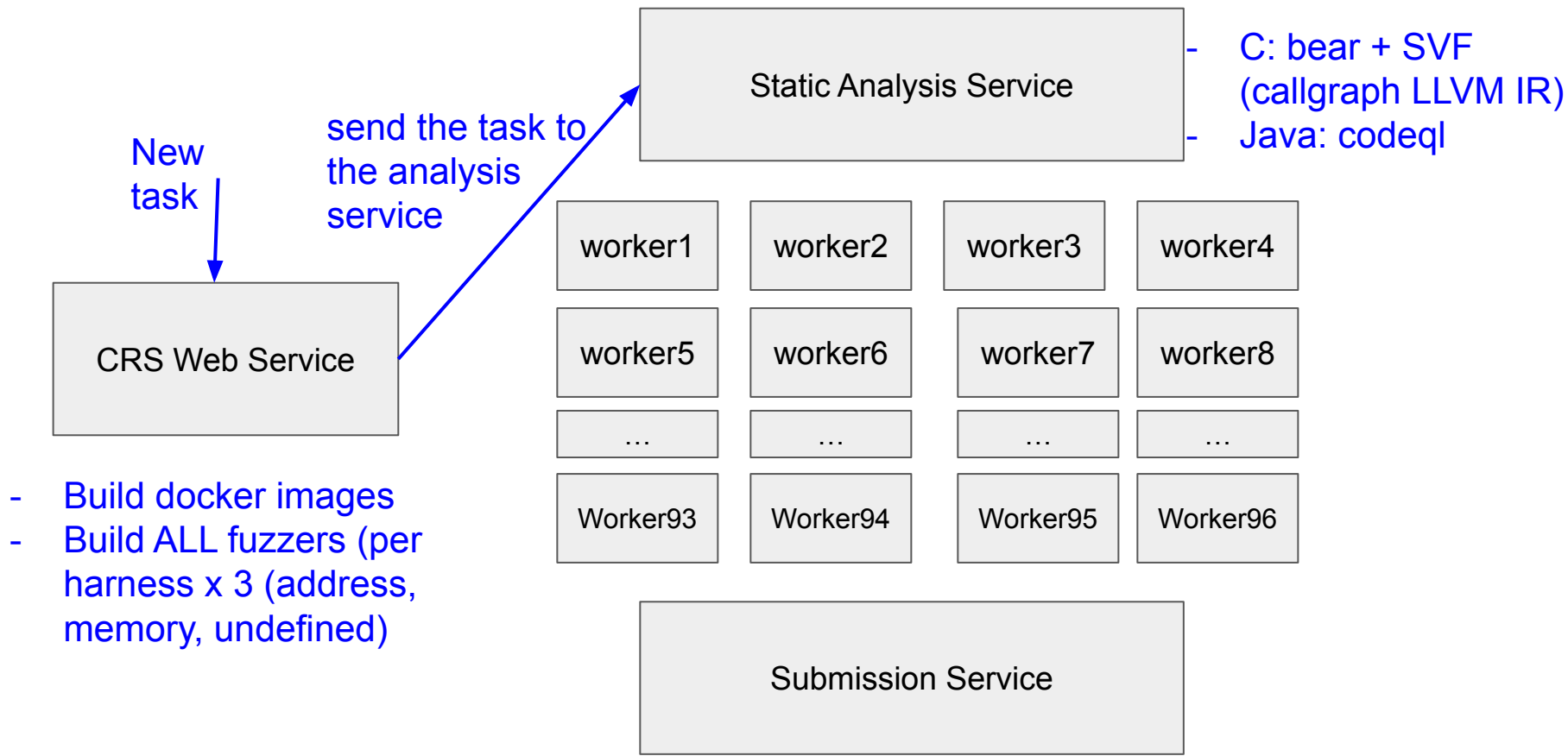
# An Overview of Our CRS



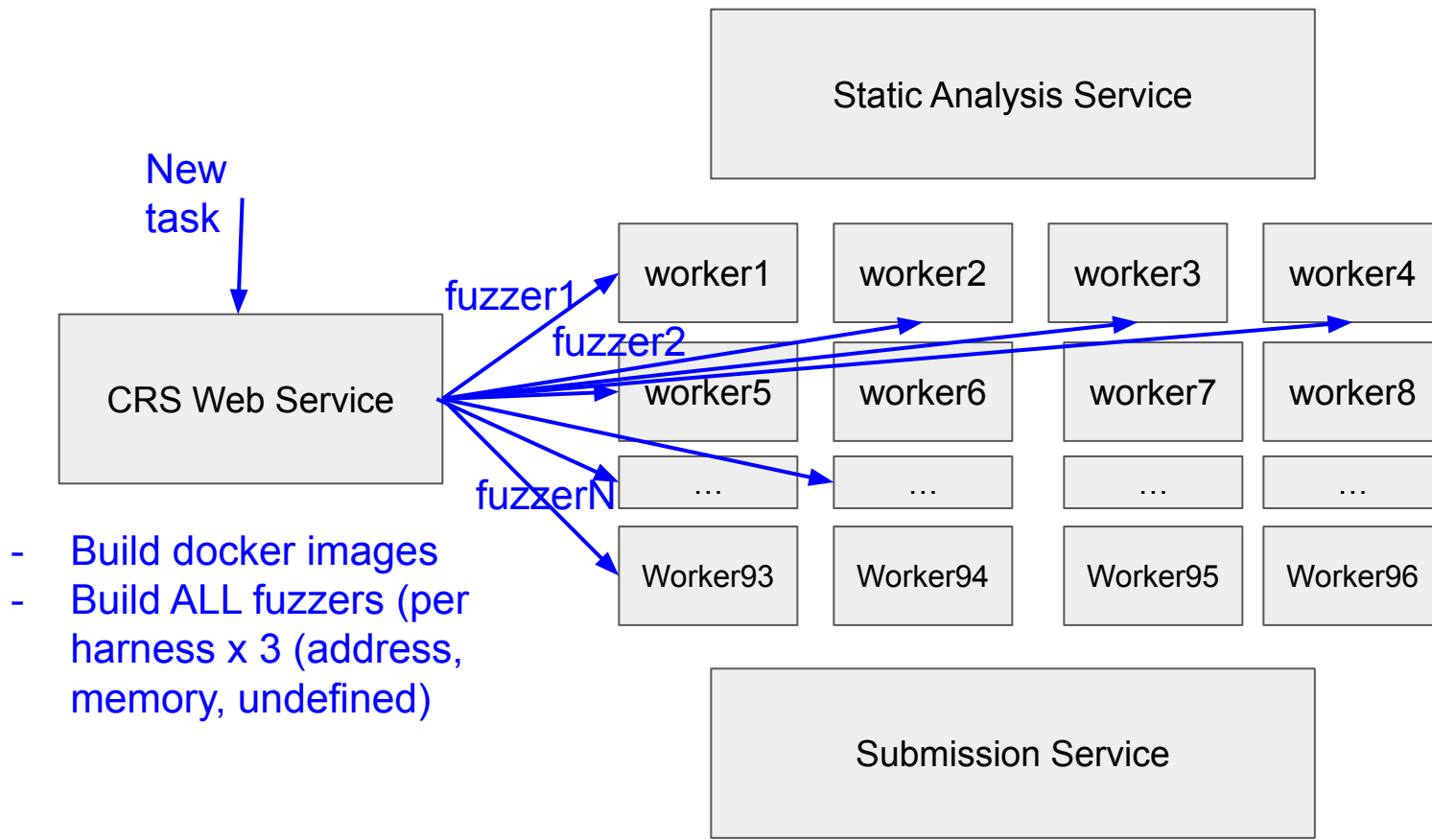
# An illustration of CRS workflow



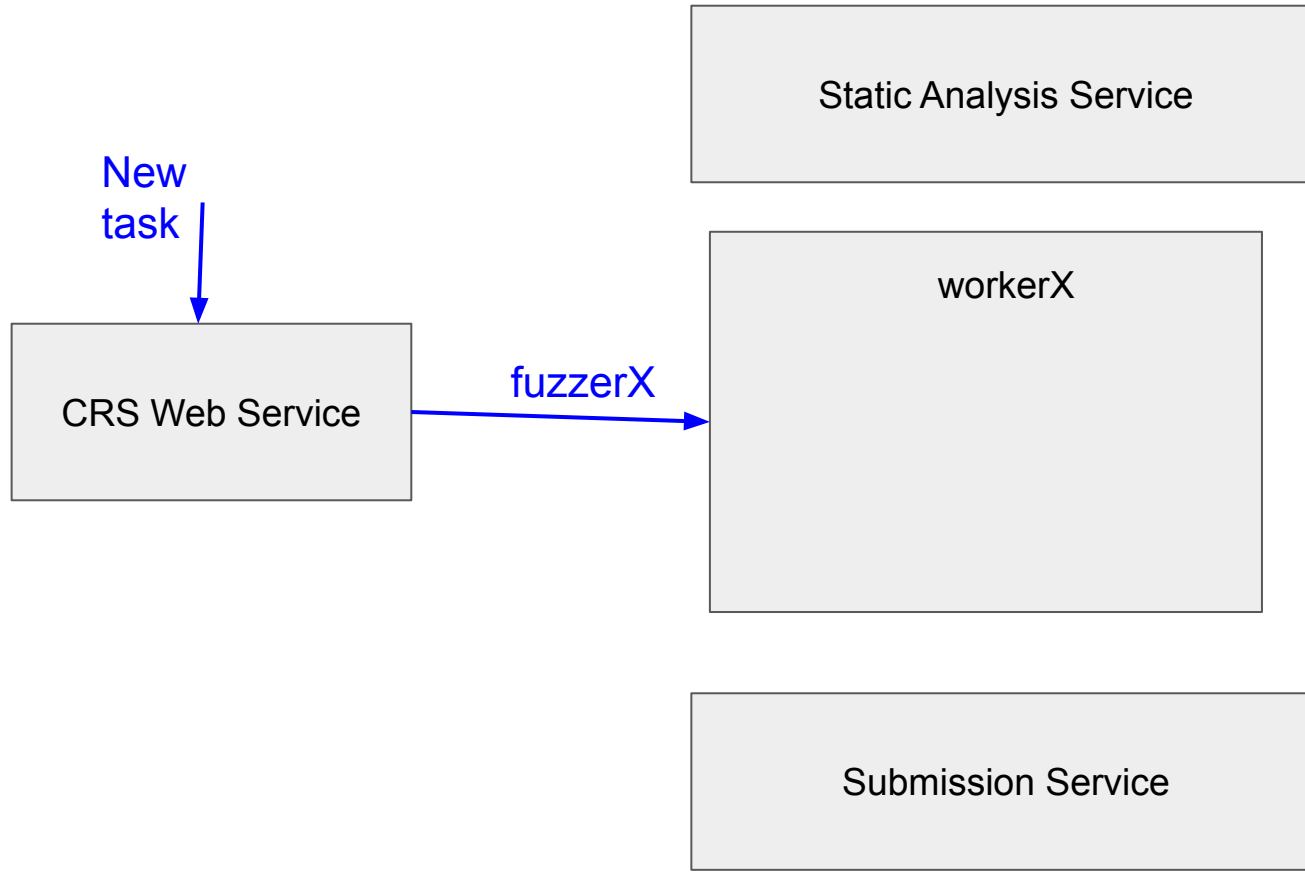
# An illustration of CRS workflow



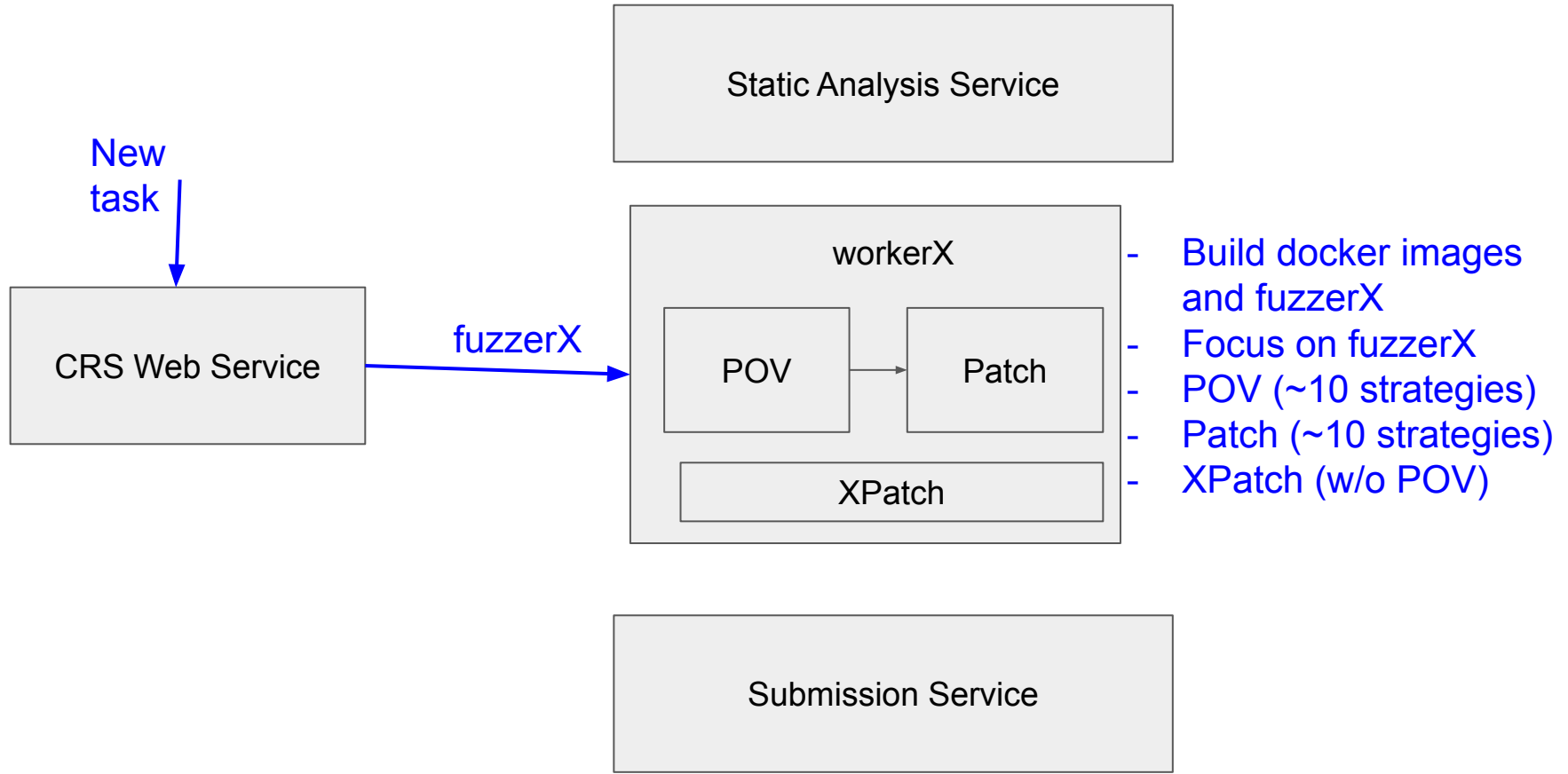
# An illustration of CRS workflow



## An illustration of CRS workflow

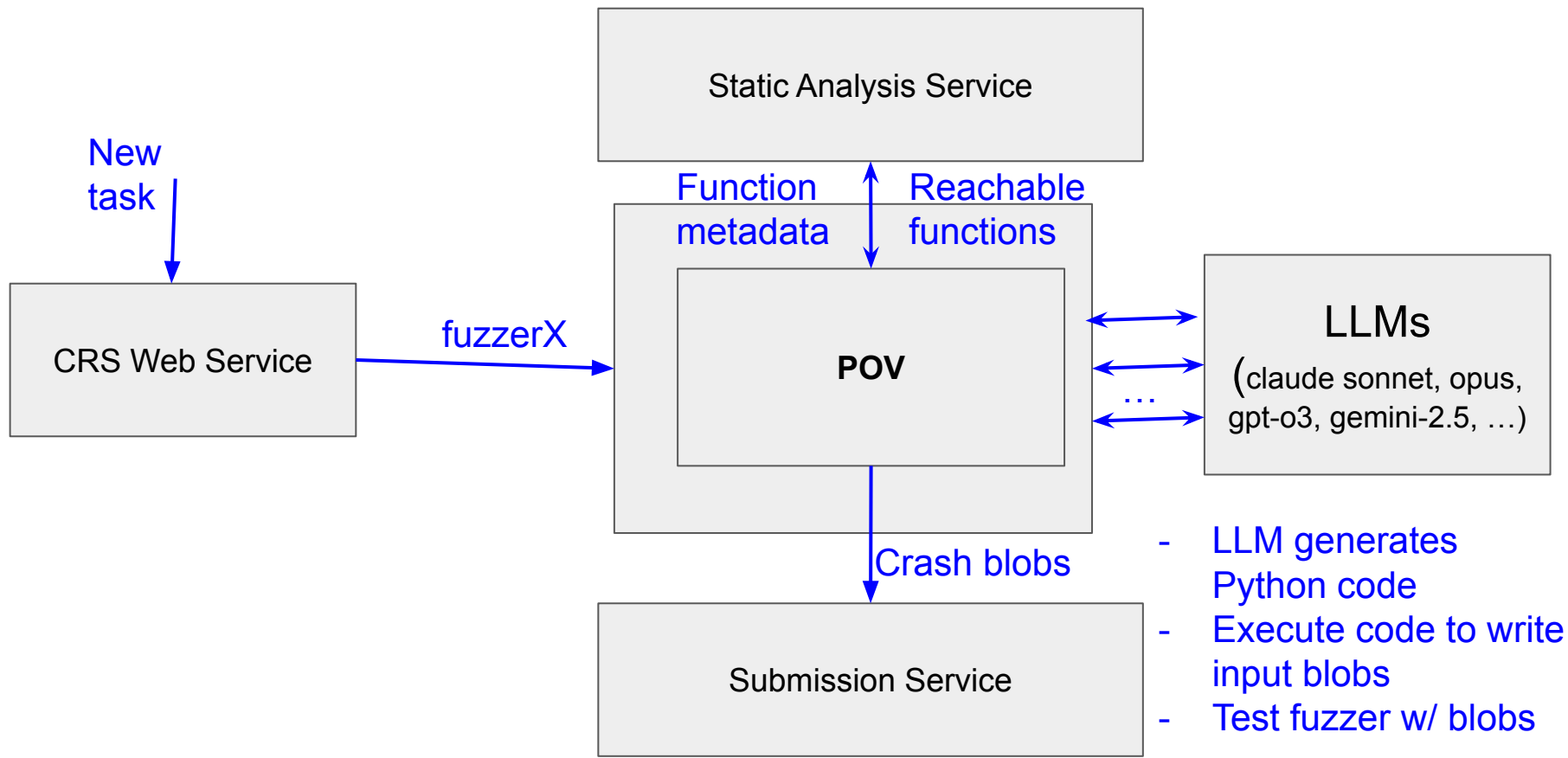


# An illustration of CRS workflow

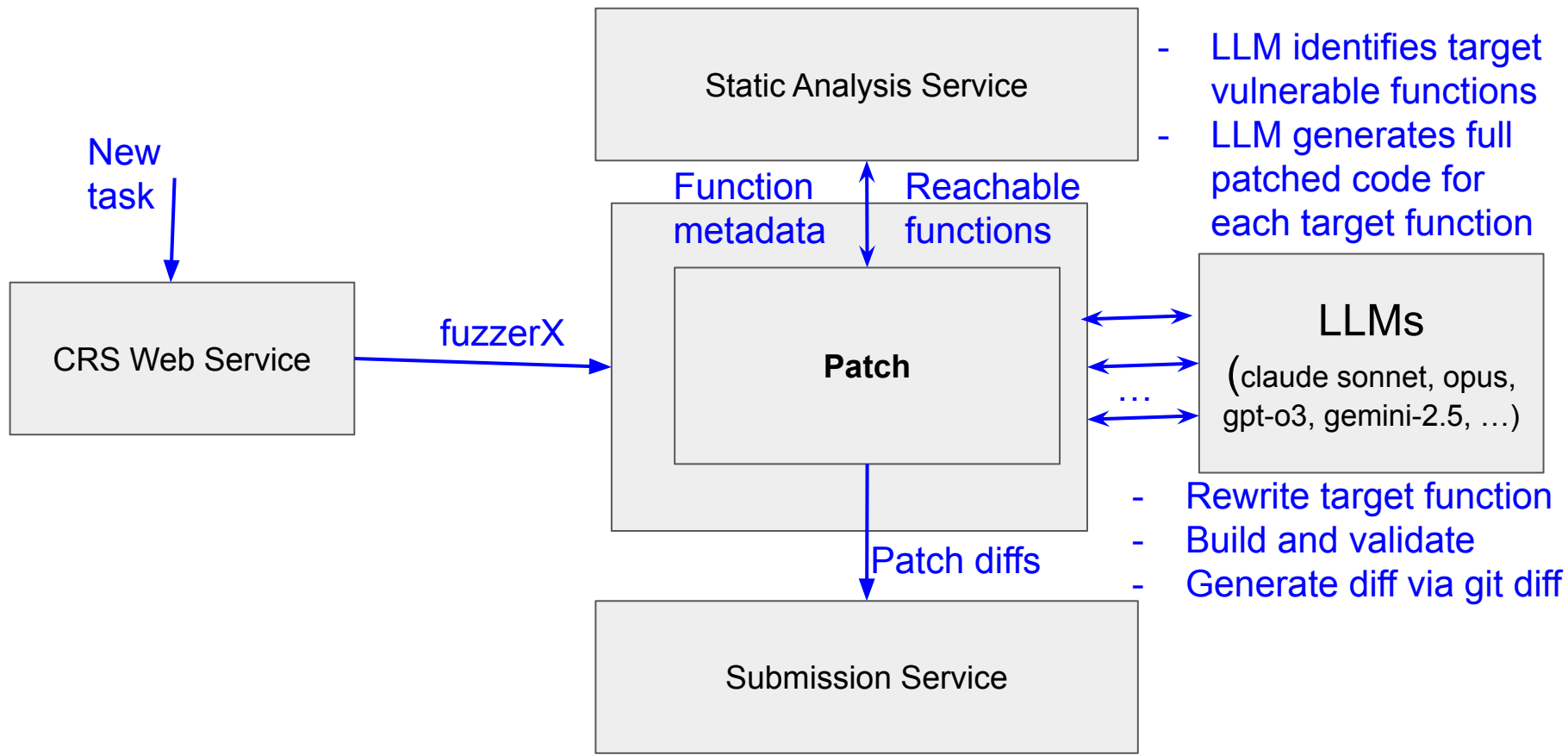




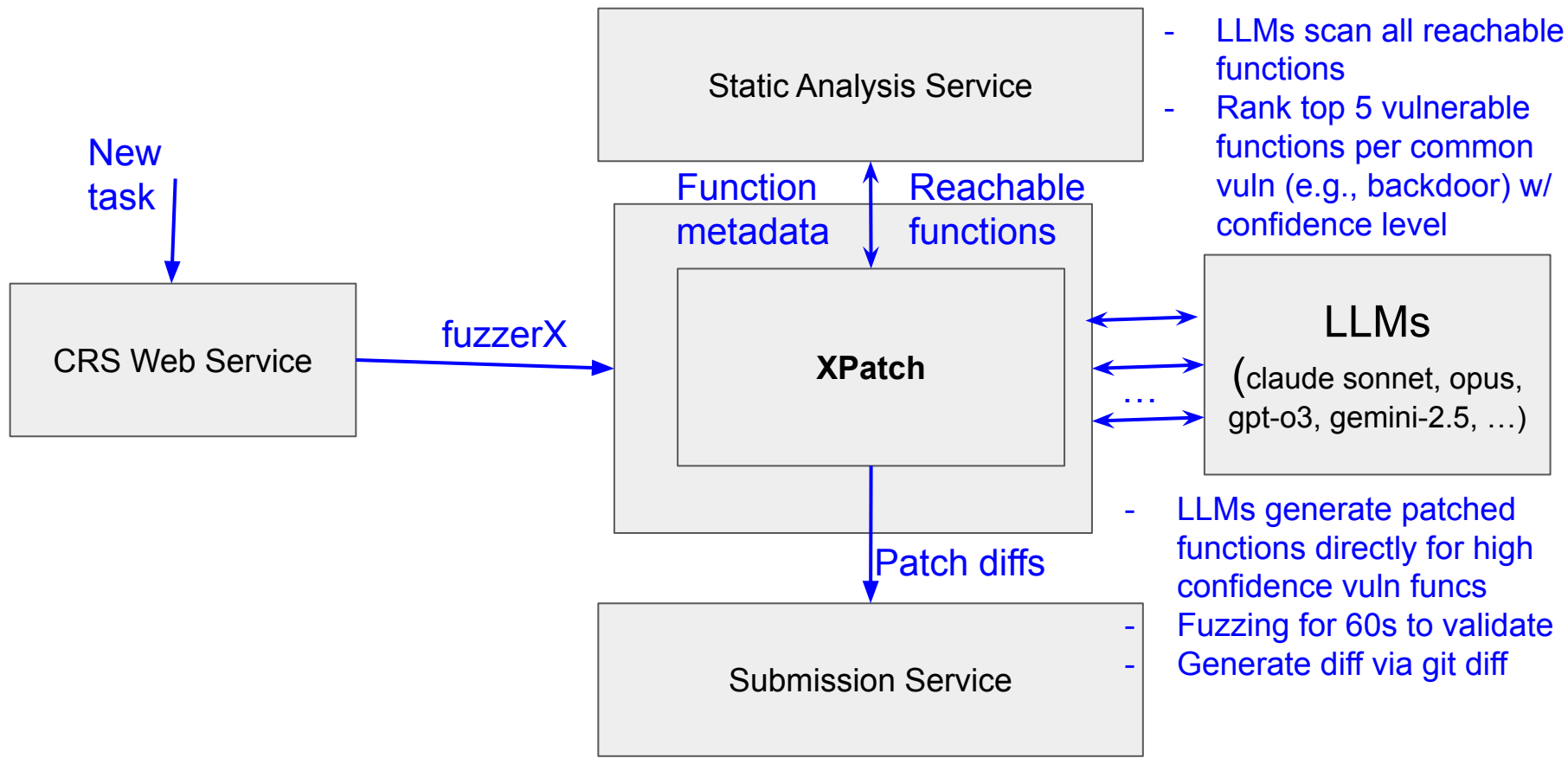
## An illustration of CRS workflow



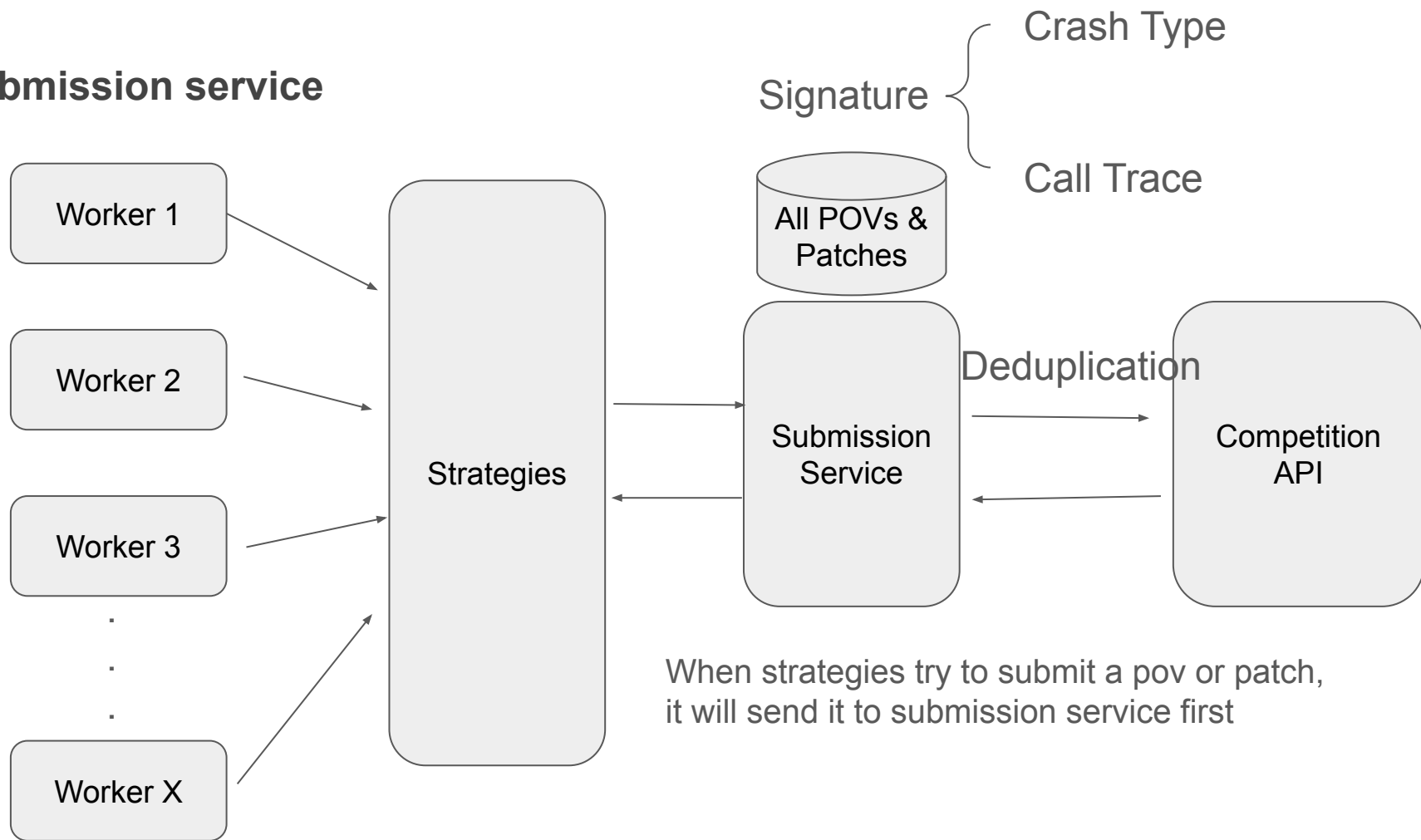
# An illustration of CRS workflow



# An illustration of CRS workflow



## Submission service

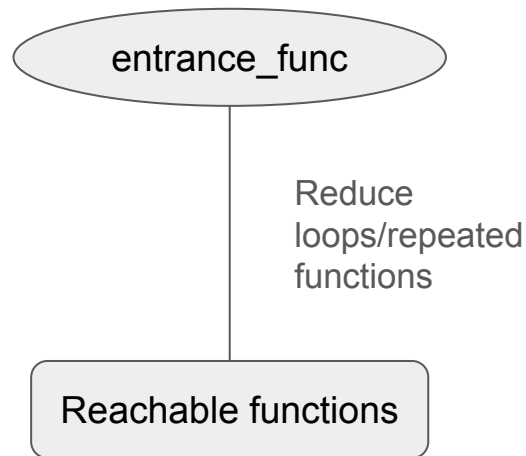


## Static analysis service

- CodeQL

Reachability Analysis

Call Path Analysis



Harness 1:

Reachable functions: func\_1/func\_2/func\_3...

Call Paths:

func\_1:{entrance\_func->func\_...->func1}


func\_2:{entrance\_func->func\_...->func2}

...

## What surprised us and what challenged us the most?

- The speed of LLM improvements is dramatic
  - LLM capabilities for discovering/patching real-world vulnerabilities, even for open models
    - GPT-3.5 -> gpt-4o -> claude-3.5 -> gpt-o1 -> gpt-o3, opus-4, gemini-2.5 ...
- LLM can often successfully generate complex working code in 1-2 attempts
  - >90% of our CRS code is generated by AI (Cursor)
  - 50 KLOC Go
  - 50 KLOC Python
- Building a reliable CRS is highly challenging
  - Testing and debugging LLM-based components
  - Rate limits, race conditions, fallback models, etc

# Specific challenges

1. Developing our CRS using multiple programming languages with concurrent strategies created race conditions and crashes, requiring extensive testing and debugging efforts
2. For full-scan challenges requiring comprehensive codebase analysis, the search space remained enormous despite employing multiple static analysis strategies and pruning techniques
3. Patch generation proved challenging as LLMs often produced limited fixes that could pass PoV tests but ed functional testing, and struggled with correct patch formatting - particularly handling line numbers and chunk diff metadata correctly
4. Long-context conversations significantly increased LLM hallucination rates and false positives, leading to CRS inefficiency and unreliable outputs during extended analysis sessions

# A race condition (revealed in Exhibition Round2)

Multiple parallel POV strategies generated input blobs in the same folder with the same name...

- Successful POV blobs are overwritten by unsuccessful ones
- Leading to **many False Positive POVs** submitted

```
pov_success, blob_path = GeneratePOV(strategy_x, llm_x)

if pov_success:
    submitPOV(blob_path)
```



# What should AI cybersecurity pursue next?

- **Further improvements of open & frontier models**
  - We plan to position our CRS as a standardized benchmark that allows researchers to test and compare the performance of various LLMs
- **Developing training/evaluation datasets for LLM cyber-reasoning capabilities**
- **More advanced PoV/Patching agents**
  - Efficiency (faster and cheaper)
  - More complex vulnerabilities (logical flaws, no harnesses, etc)
  - Patch Validation, current validation is just pov & functionality tests. We need more robust patch validation strategy.
- **System-support for building AI-based CRS**
  - Testing and debugging tools

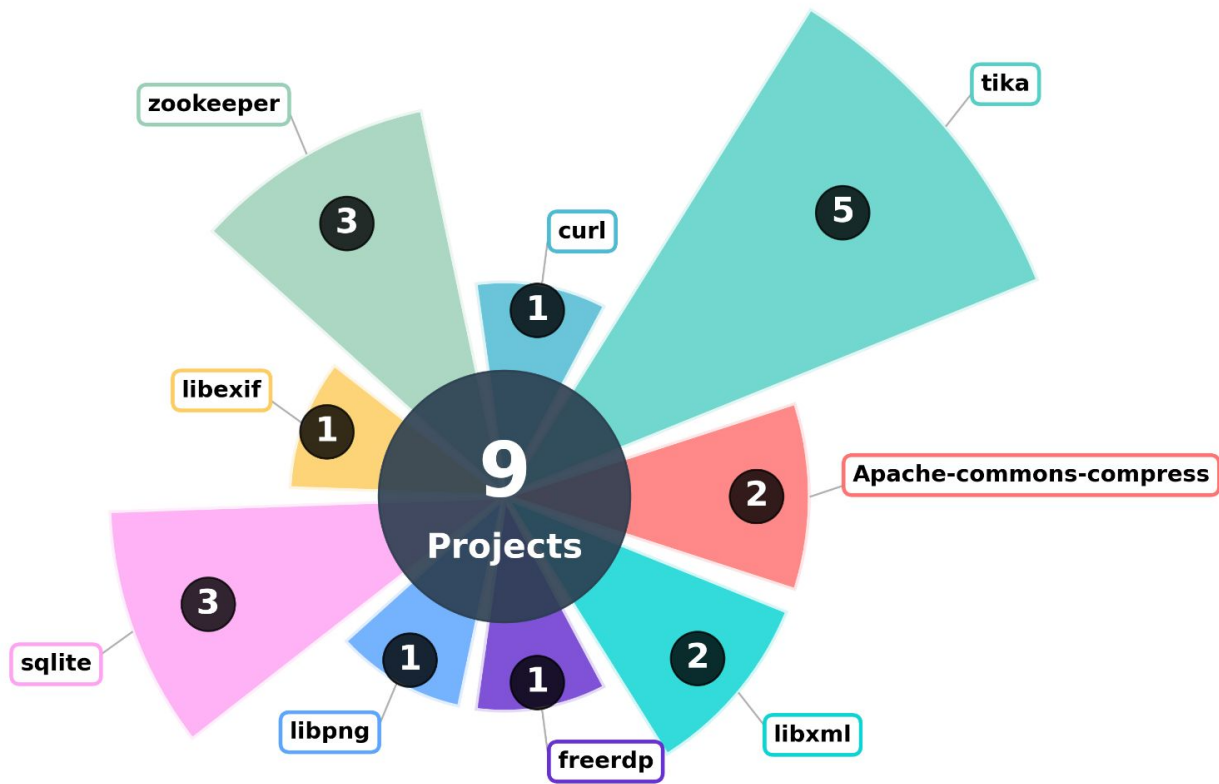
# Open Source vs. Flagship Models

- Much effort was put towards working with LLAMA, QWEN, or Deepseek for steps in our pipeline
- Evaluations could not justify the switch
- Time required to fine tune models was not realistic

# NGINX Test Environment Results (Locating Only)

Model	CP V1	CP V2	CP V3	CP V4	CP V5	CP V8	CP V9	CP V10	CP V11	CP V12	CP V13	CP V14	CP V15	CP V17
(For Comparison) Gemini 2.0 Flash	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓
Deep Seek R1	✗	✓	✗	✗	✓	✓	✗	✗	✓	✗	✓	✗	✓	✓
LLAMA 3.1 70b	✗	✓	✗	✗	✓	✓	✗	✓	✓	✗	✓	✗	✗	✓
LLAMA 3.3 70b	✓	✓	✗	✗	✓	✓	✗	✓	✓	✗	✓	✗	✗	✓



# Model & Challenges Evaluation by Our CRS



Challenge Distribution

- 9 Projects
- 19 Delta-Scan Challenges
- 10+ CWEs
- C & Java

# Flagship Model & Challenges Evaluation by Our CRS

Model Name	POV 	Patch 	Average Time (Include task building)
Claude-Sonnet-4	18/19	19/19	22:03
Claude-Opus-4	18/19	19/19	32:58
Claude-Sonnet-3.7	18/19	19/19	25:17
Claude-Sonnet-3.5	15/19	11/19	28:12
GPT-4.1	18/19	19/19	18:29
GPT-o4-mini	18/19	15/19	19:51
GPT-o3-pro	18/19	16/19	39:58
GPT-4o	12/19	7/19	20:43

# Open Source

**Full CRS** (identical to what's used in the aixcc final round)

<https://github.com/o2lab/afc-crs-all-you-need-is-a-fuzzing-brain>

**Local version** (runs on a single VM w/ docker image):

- <https://github.com/o2lab/afc-crs-all-you-need-is-a-fuzzing-brain/tree/jeff>
- 

**Semi-final CRS:** <https://github.com/o2lab/asc-crs-all-you-need-is-a-fuzzing-brain>