OSS-CRS: Bug-Finding and Remediation for the LLM era

Andrew Chin

SSLab @ Georgia Institute of Technology

Team Atlanta

OpenSSF Community Day

NORTH AMERICA 2026

Bug-finding

Triaging

Patching

DARPA + ARPA-H

AIxCC
AI CYBER CHALLENGE

PATCHING CRITICAL INFRASTRUCTURE

DARPA’s AI Cyber Challenge
Announced on 08/2023

DARPA + ARPA-H

AIxCC
AI CYBER CHALLENGE

WHAT IS AIxCC?

➔A competition that rewards autonomous systems that *find* and *patch* vulnerabilities in source code — **Cyber Reasoning Systems (CRS)**.

➔The challenges are well-known open-source projects.

➔The vulnerabilities are realistic or real.

➔Patching is worth more than finding.

➔Code and data will be released open source.

black hat

AUGUST 2023

OPEN TRACK AND SMALL BUSINESS TRACK SUBMISSIONS

DEFCON

AUGUST 2024

SEMIFINAL COMPETITION

Top 7 teams $2 million each

DEFCON

AUGUST 2025

FINAL COMPETITION

Winners announced

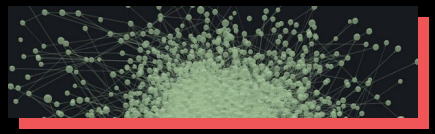
1ST: $4 MILLION

2ND: $3 MILLION

3RD: $1.5 MILLION

Google ANTHROP\C OpenAI Microsoft THE LINUX FOUNDATION OpenSSF OPEN SOURCE SECURITY FOUNDATION

What counts for semifinals?



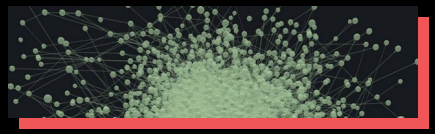
Proof-Of-Vulnerability (POV)

➔Input data to reproduce vulnerability crash in harness

PATCH

➔Unified diff source code fix for vulnerabilities

What counts for finals?

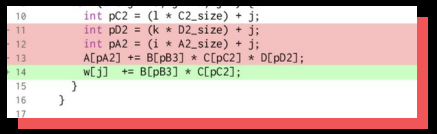


Proof-Of-Vulnerability (POV)

➔Input data to reproduce vulnerability crash in harness

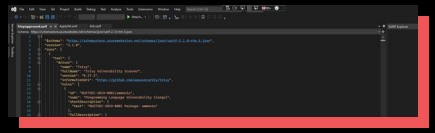
PATCH

➔Unified diff source code fix for vulnerabilities



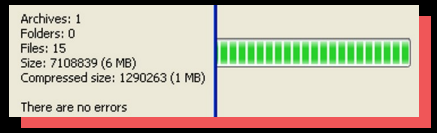
DELTA SCAN

➔Challenge analyzing base code plus applied diff changes



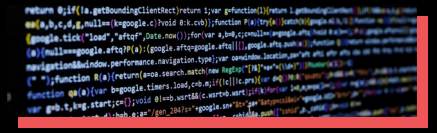
SARIF Assessment

➔Structured reporting format for vulnerability details



BUNDLE

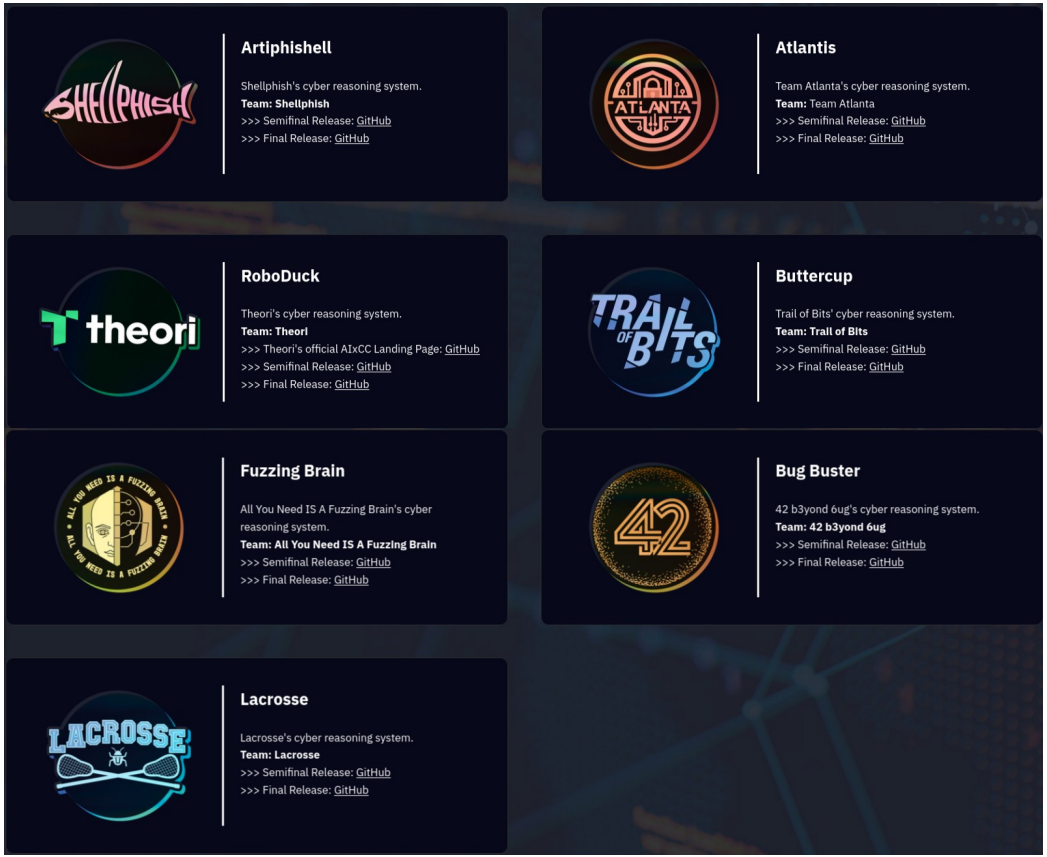
➔Grouping of related PoV, patch, and SARIF submissions



FULL SCAN

➔Challenge analyzing entire code base

Open Sourced!



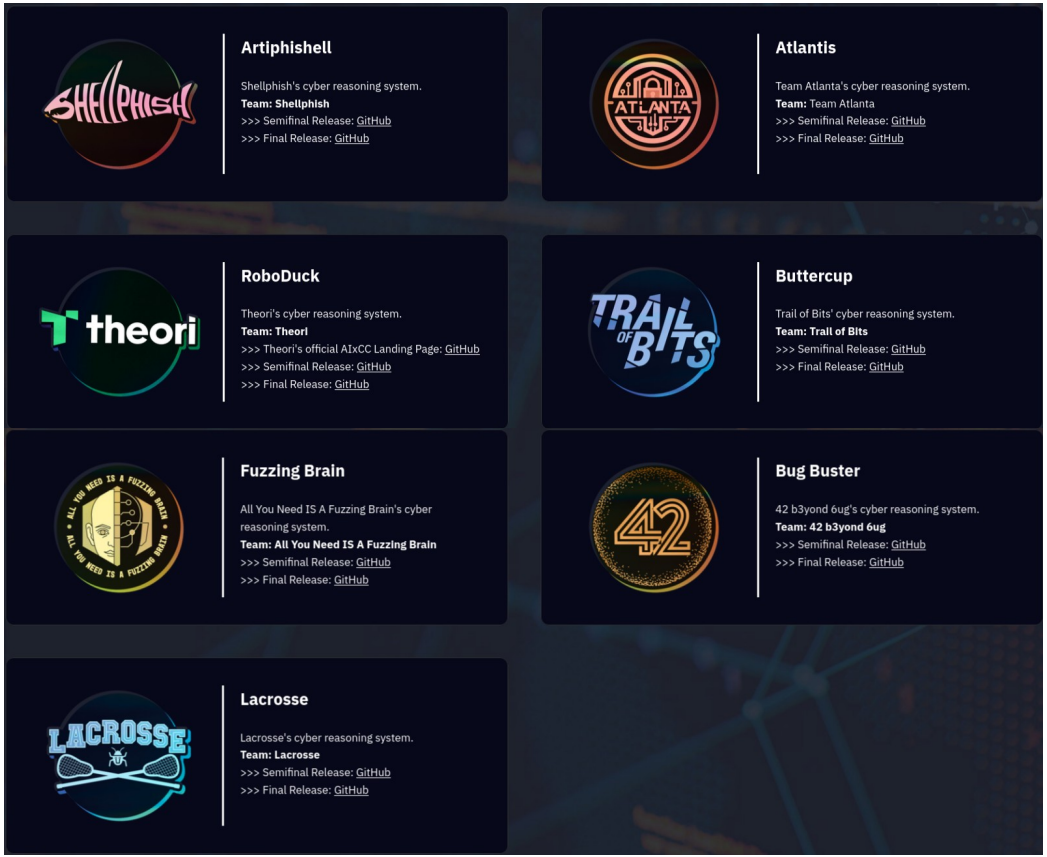
Open Sourced! But largely unmaintained...

No Activity

Archived

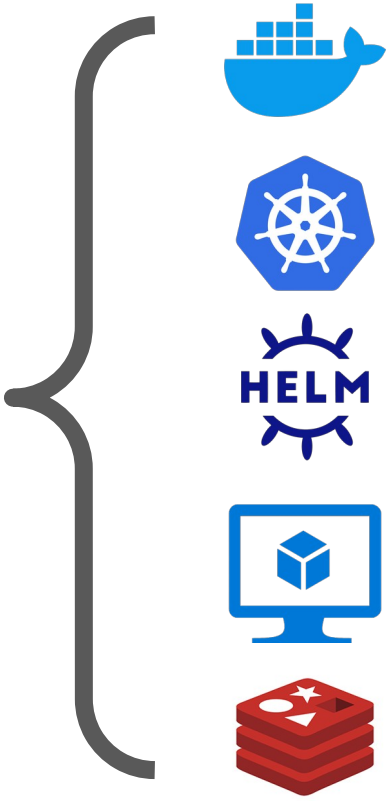
Active

No Activity

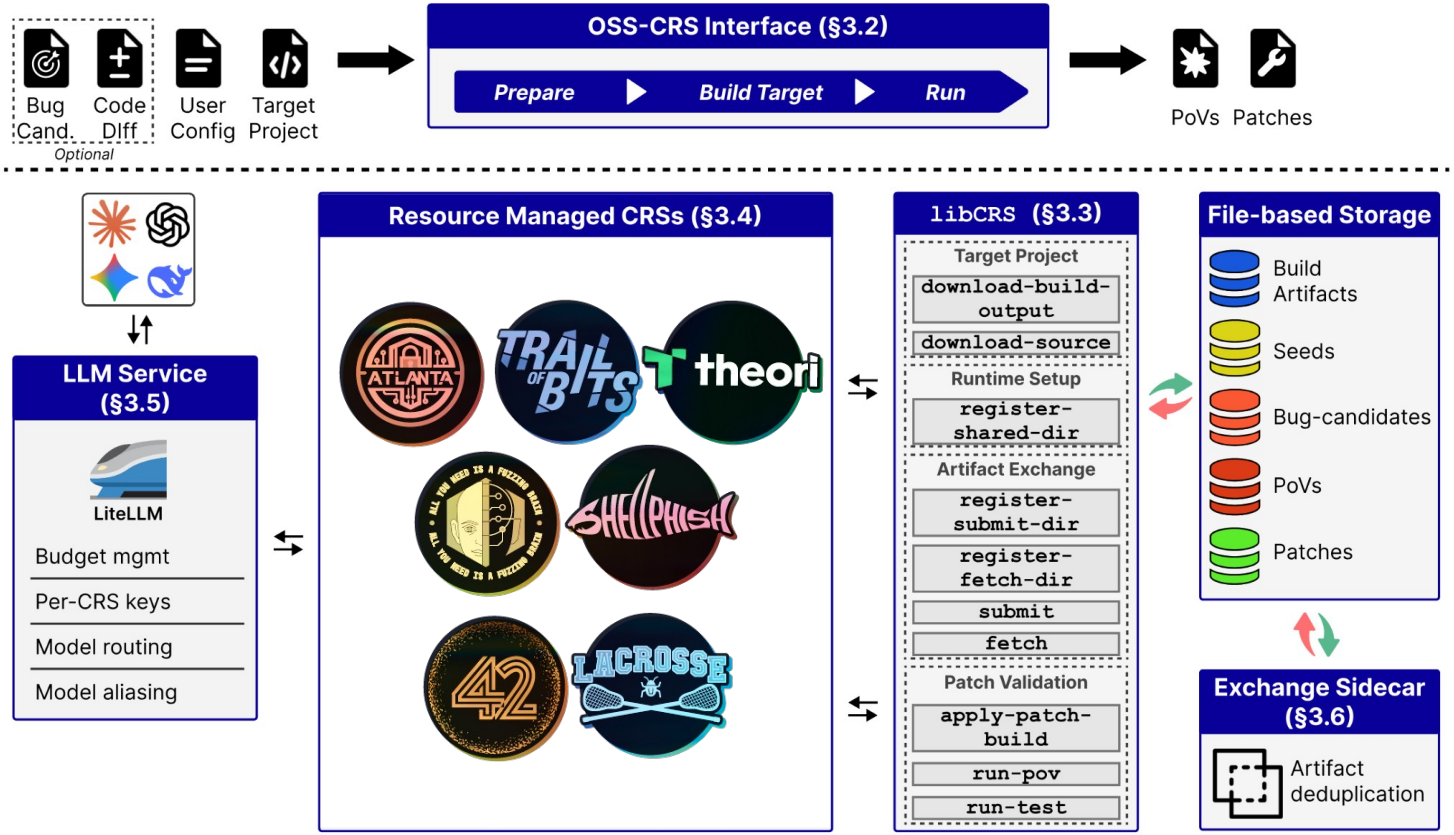


Running All Finalist CRSs

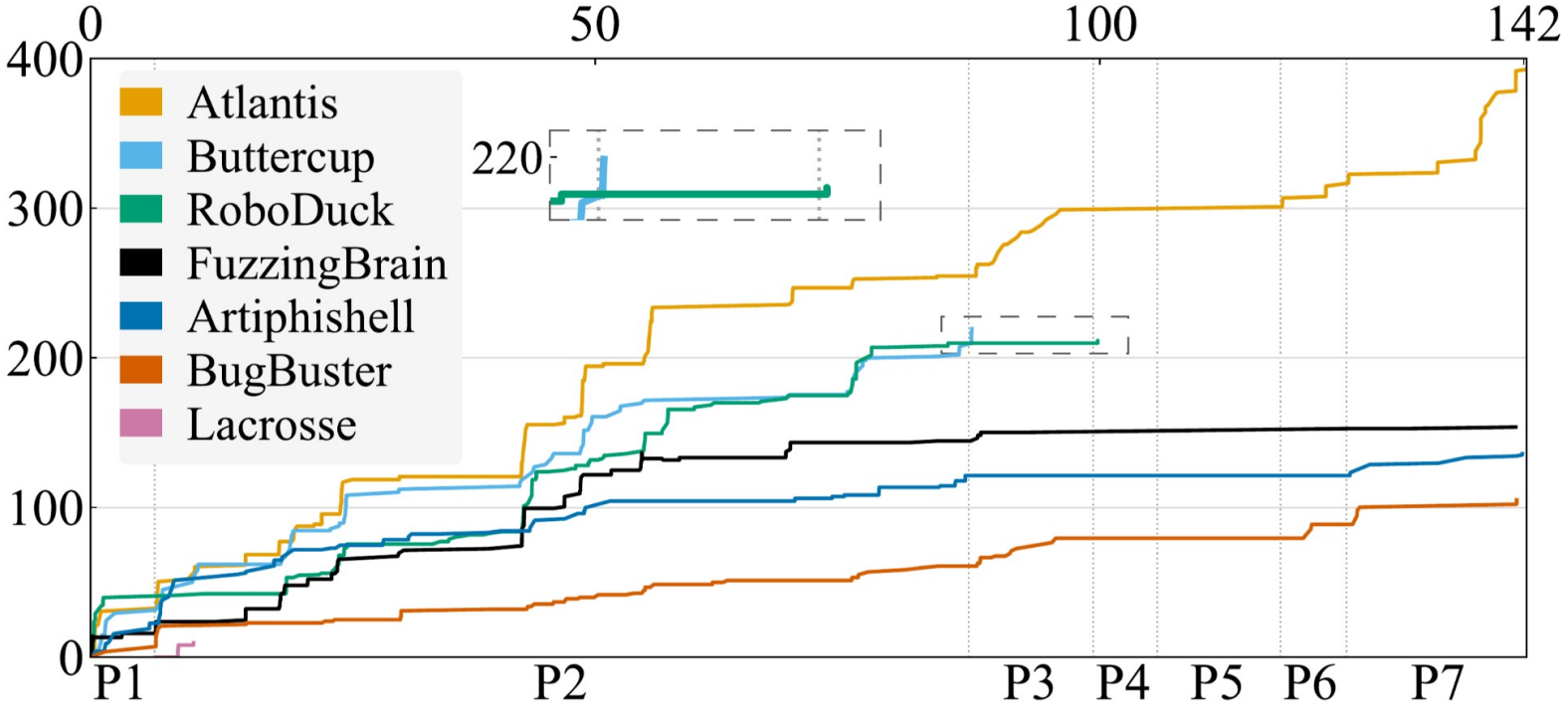
- ✗ Cloud Lock-In
- ✗ Infrastructure Duplication
- ✗ Monolithic Design



OSS-CRS: Open CRS Framework



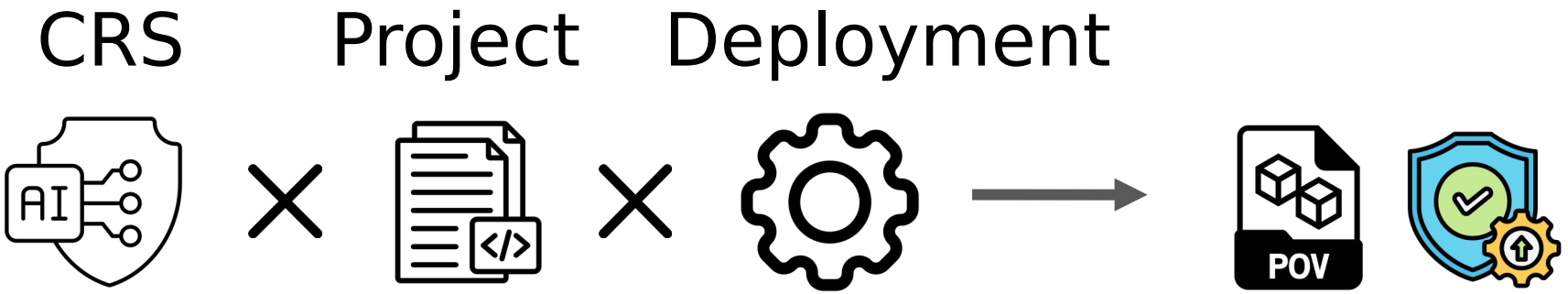
AIxCC: Score Over Time

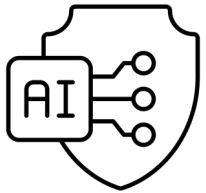


OSS-CRS Features

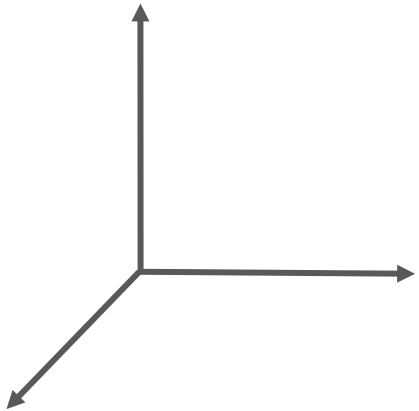
- ➔ Compatible with OSS projects integrated with **OSS-Fuzz**
- ➔ Drives both **bug-finding** and **patching** CRSs (and more!)
- ➔ Flexible **LLM** endpoint configurations
- ➔ **Locally** deployable (run it on your laptop!)
- ➔ Resource management and LLM budgeting as first class

OSS-CRS at a high level



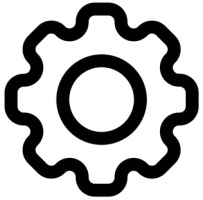


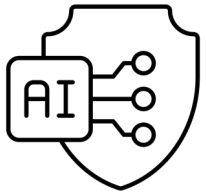
CRSs



Projects

Deployments





Researchers

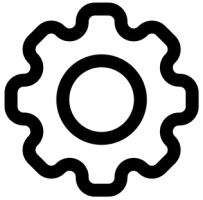
CRSs

Projects

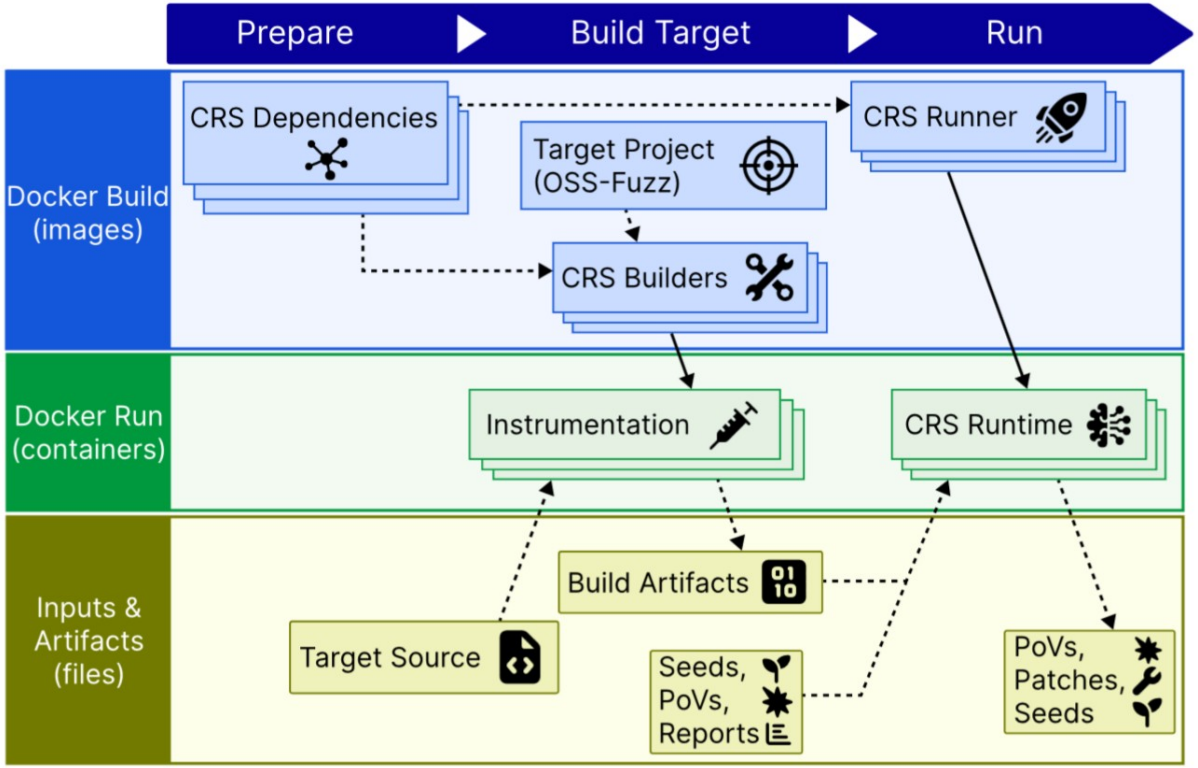
Maintainers

Security Engineers

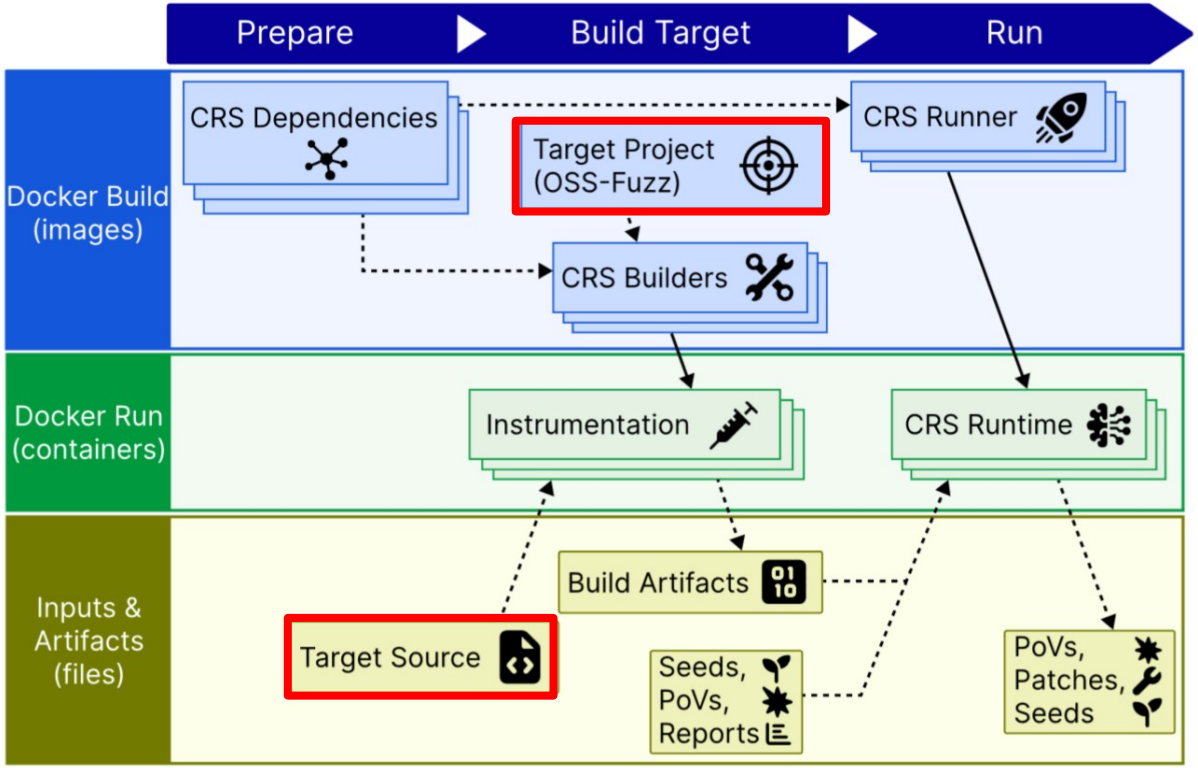
Deployments



Stages


Prepare
Build Target
Run
Docker Build (images)
CRS Dependencies
Target Project (OSS-Fuzz)
CRS Runner
CRS Builders
Docker Run (containers)
Instrumentation
CRS Runtime
Inputs & Artifacts (files)
Build Artifacts
Target Source
Seeds, PoVs, Reports
PoVs, Patches, Seeds

@Project Maintainers



OSS-Fuzz

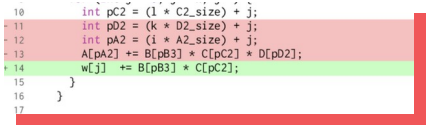
- > 1000 OSS projects in the ecosystem
- Standardizes compilation and the execution environment
- Fuzz testing continuously done by Google (Clusterfuzz)
- Threat model defined by “harnesses”

OSS-Fuzz Format

- Dockerfile – container environment for building the project
- build.sh – script for compiling the project
- Fuzz harnesses – entrypoint that passes data to public APIs
- (Optional) run_tests.sh – functionality tests

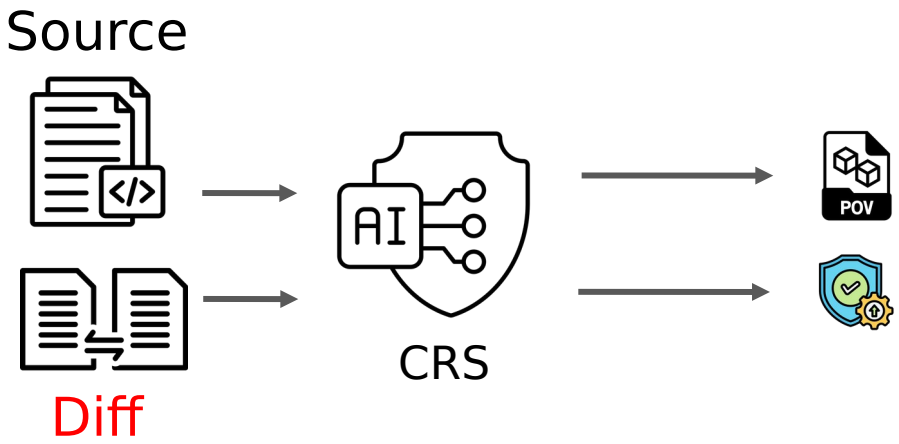
```cpp
#include <stdint.h>
#include <stdlib.h>

#include <memory>

#include <turbojpeg.h>


extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {
    tjhandle jpegDecompressor = tjInitDecompress();

    int width, height, subsamp, colorspace;
    int res = tjDecompressHeader3(
        jpegDecompressor, data, size, &width, &height, &subsamp, &colorspace);

    // Bail out if decompressing the headers failed, the width or height is 0,
    // or the image is too large (avoids slowing down too much). Cast to size_t to
    // avoid overflows on the multiplication
    if (res != 0 || width == 0 || height == 0 || ((size_t)width * height > (1024 * 1024))) {
        tjDestroy(jpegDecompressor);
        return 0;
    }

    std::unique_ptr<unsigned char[]> buf(new unsigned char[width * height * 3]);
    tjDecompress2(
        jpegDecompressor, data, size, buf.get(), width, 0, height, TJPF_RGB, 0);

    tjDestroy(jpegDecompressor);

    return 0;
}
```

Delta Scan Mode For CI/CD

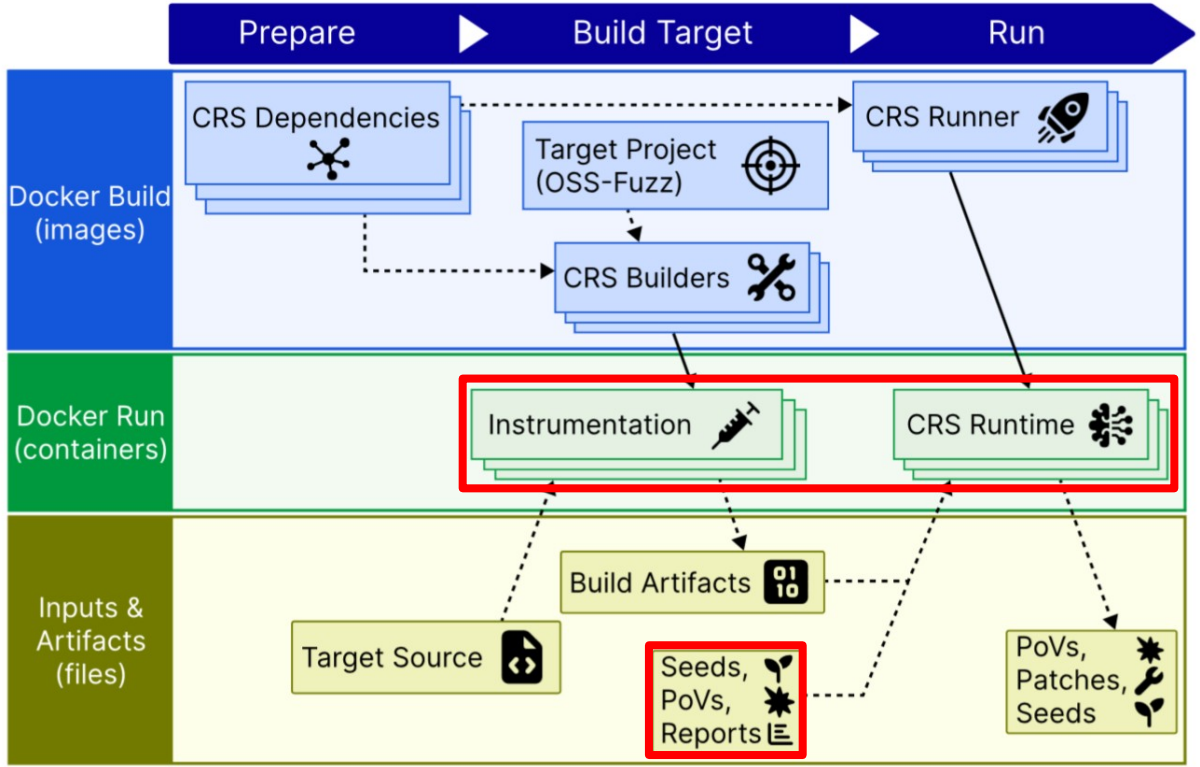


DELTA SCAN

Challenge analyzing base code plus applied diff changes



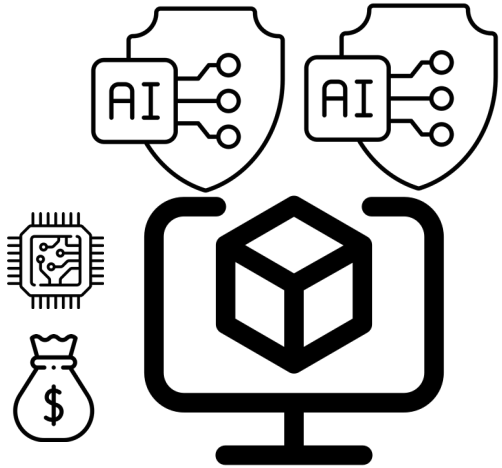
@Security Engineers



Using OSS-CRS

The operator of OSS-CRS can decide:

- Which CRSs to run
- How much compute resources to give to each CRS
- How much LLM budget to give to each CRS
- Which models and endpoints to route LLM requests to
- How long to run the CRSs for
- What project (and harness) to run the CRSs against
- Which vulnerability to fix (for patching CRSs)



Operator Configuration

For a CRS's build and run phase,
OSS-CRS can apply resource limitations

- CPU set
- Memory
- LLM budget and rate limits

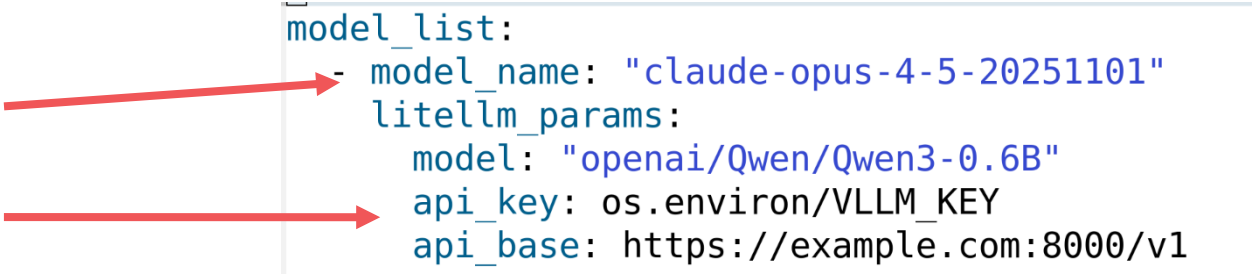
compose.yaml

```yaml
# `oss-crs-infra` must be required
oss_crs_infra:
  cpuset: "0-3"
  memory:  "16G"

crs-libfuzzer:
  cpuset: "8-11"
  memory:  "16G"

multilang:
  cpuset: "4-7"
  memory:  "16G"
  llm_budget: 100 # 100 dollars


llm_config:
  litellm_config: ./example/multilang/litellm-config.yaml
```

LLM Configuration

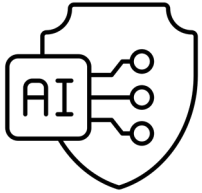
Operators can define a custom configuration for LiteLLM proxy

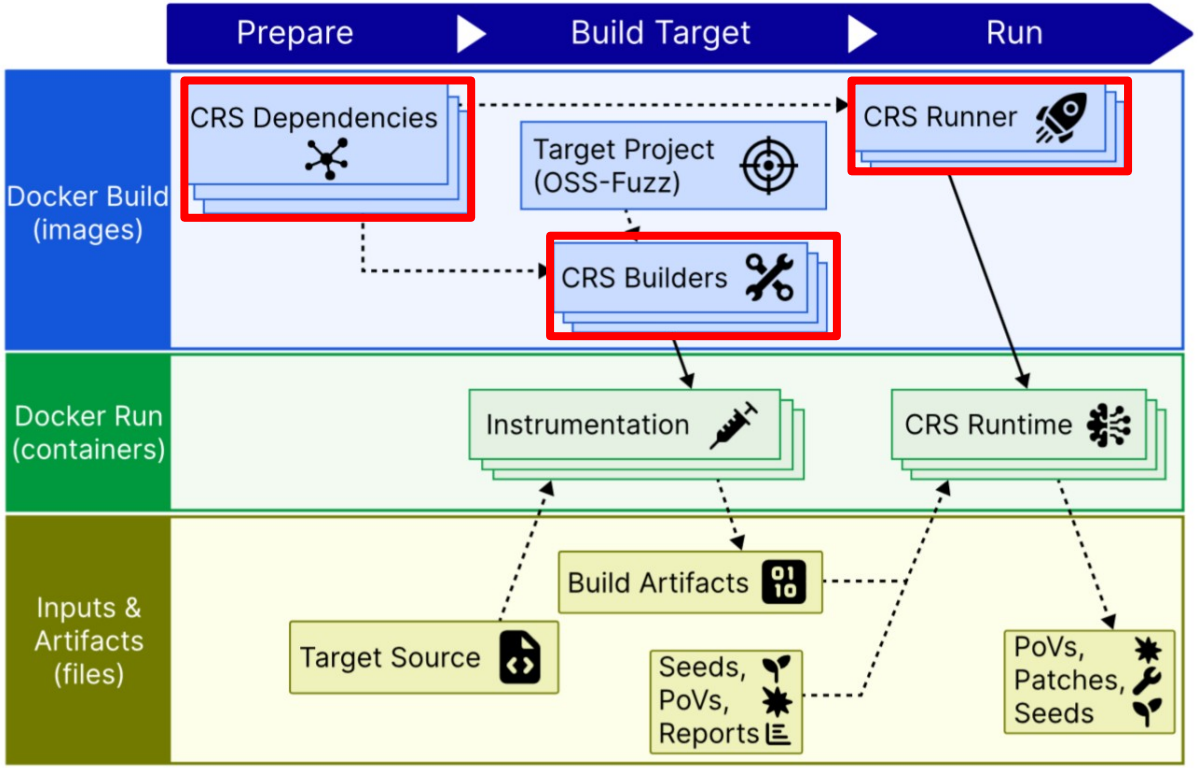
- Alias LLM model names
- Bring your own API keys



CRSs should re-route their LLM requests to an OSS-CRS-provided base URL and key

@Researchers





CRS Integration Configuration

Register CRS into OSS-CRS

[ossf/oss-crs: registry/atlantis-java-main.yaml]

```yaml
name: atlantis-java-main
type:
  - bug-finding
source:
  url: https://github.com/Team-Atlanta/atlantis-java-snapsho
  ref: main
```

Declare CRS containers and requirements

[Team-Atlanta/atlantis-java-main: oss-crs/crs.yaml]

```yaml
name: atlantis-java-main

target_build_phase:
  # additional containers can be specified
  build:
    dockerfile: oss-crs/builder.Dockerfile

crs_run_phase:
  run:
    dockerfile: oss-crs/runner.Dockerfile

allowed_llms:
- gpt-5
- o4-mini
- o3
- gpt-4.1
```

libCRS: helpers and interface for CRSs

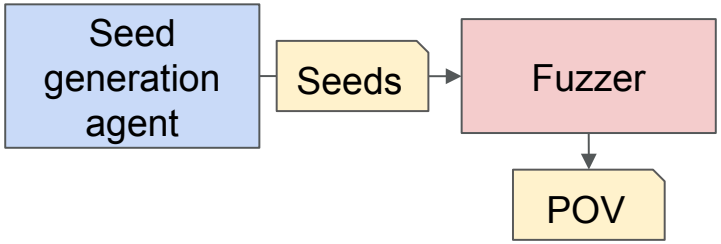
Artifact transfer

- Build outputs – syncing build artifacts between stages
- Submitting findings – PoVs, patches
- Fetching auxiliary data – seeds, bug candidates, PoVs, patches

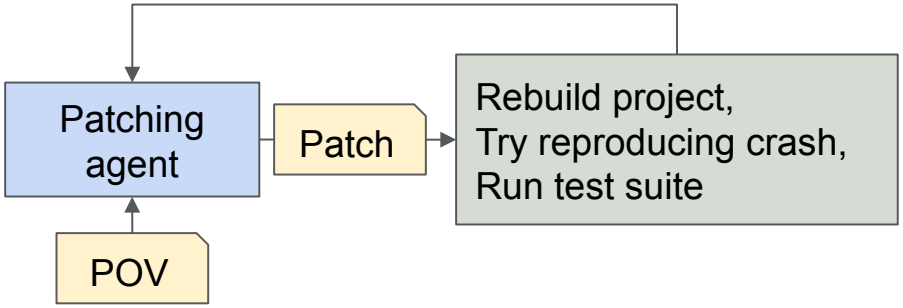
Helpers for common CRS actions

- Rebuilding the project given a patch – can use custom CRS builder!
- Run PoV to check if buggy behavior reproduces
- Run functionality tests

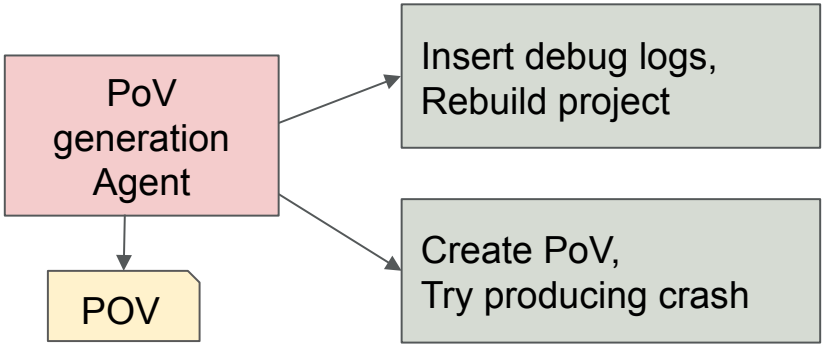
Common CRS Flows



Seed generation + fuzzer



Patching agent + rebuild



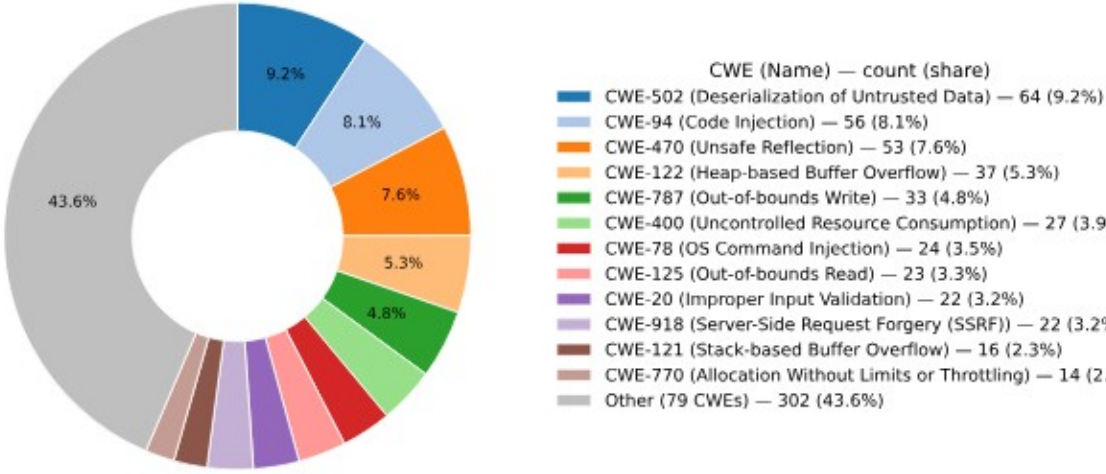
Pure agentic PoV generation

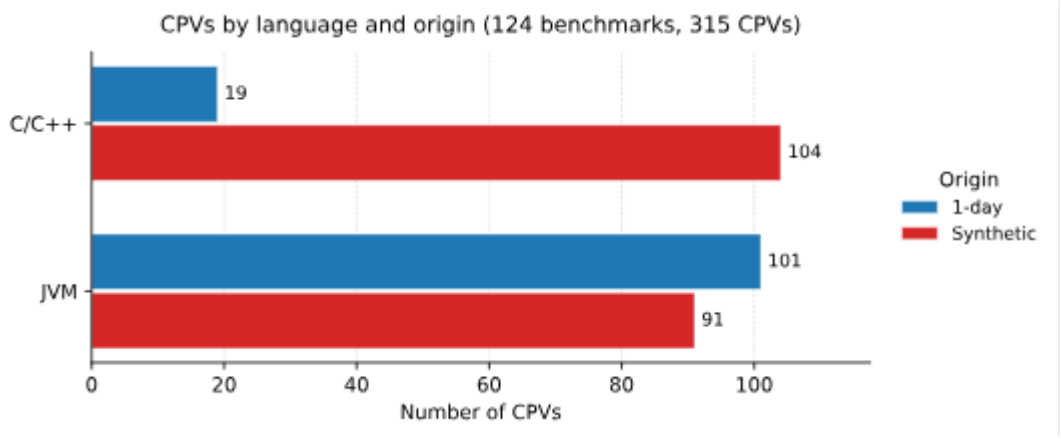
CRSBench

Benchmarking and testing CRSs

- Use any CRS integrated into OSS-CRS
- Faster evaluation using build snapshots and selective unit testing
- Dataset made from AI Cyber Challenge problems and Team Atlanta internal benchmarks

oss-crs.openssf.org/crsbench





Putting Everything Together

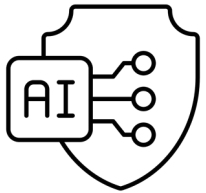
```
# Build CRS Docker images
uv run oss-crs prepare \
    --compose-file example/crs-bug-finding-codex/compose.yaml

# Build target project image and harnesses
uv run oss-crs build-target \
    --compose-file example/crs-bug-finding-codex/compose.yaml \
    --fuzz-proj-path ~/post/CRSBench/benchmarks/asc-nginx-delta-01

# Run CRS against a harness (entrypoint)
uv run oss-crs run \
    --compose-file example/crs-bug-finding-codex/compose.yaml \
    --fuzz-proj-path ~/post/CRSBench/benchmarks/asc-nginx-delta-01 \
    --diff ~/post/CRSBench/benchmarks/asc-nginx-delta-01/.aixcc/ref.diff \
    --target-harness pov_harness
```

Compose selects which CRS to run

“Project” dir with harness and Docker env

Harness is the entrypoint to public API

Get Involved!



Researchers

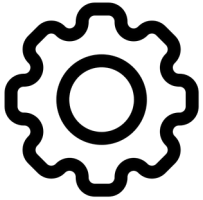
CRSs

Security Engineers

Projects

Deployments

Maintainers



OSS-CRS Links

https://oss-crs.openssf.org/

https://github.com/ossf/oss-crs

https://openssf.slack.com/archives/C09FQLYH1RD

(All of these links can be found on https://openssf.org/projects/oss-crs/)

Also check out Team Atlanta's website and blog at https://team-atlanta.github.io/

Finally, you can find my contacts on my webpage at https://achin.ca

Thank You!

OpenSSF Community Day

NORTH AMERICA 2026