

OPEN SOURCE
SecurityCon
NORTH AMERICA

OSS-CRS: Open Framework for Cyber Reasoning Systems

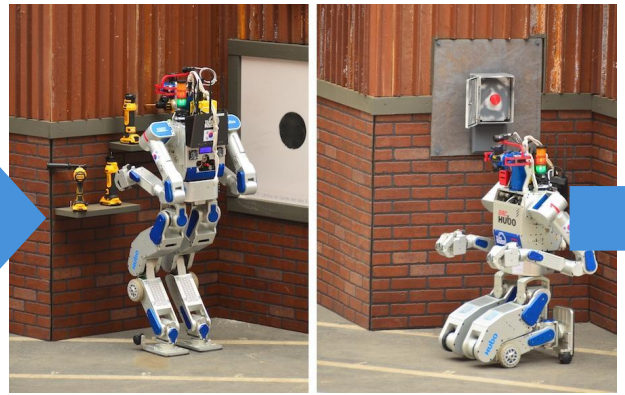
Andrew Chin
Georgia Institute of Technology

DARPA Grand Challenge

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY



2004-2007 (Autonomous Vehicle)



2012-2015 (Robotics Challenge)



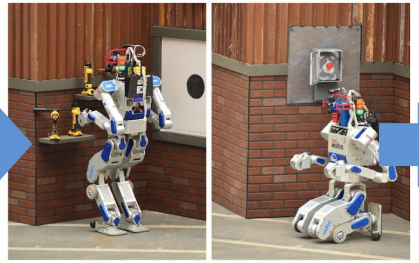
2014-2016 (Cyber Grand Challenge)

DARPA Grand Challenge

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY



2004-2007
(Aut[illegible]mous Vehicle)



2012-2015
(Robotics Challenge)

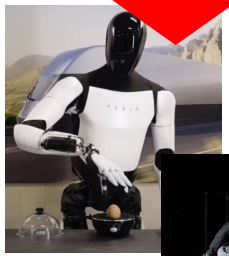


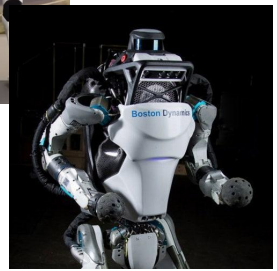
2014-2016
(Cyber Grand Chall[illegible])

20+ years









Mayhem by ForAllSecure



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

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.



Artiphishell

Shellphish's cyber reasoning system.
Team: Shellphish
>>> Semifinal Release: GitHub
>>> Final Release: GitHub



Atlantis

Team Atlanta's cyber reasoning system.
Team: Team Atlanta
>>> Semifinal Release: GitHub
>>> Final Release: GitHub



RoboDuck

Theori's cyber reasoning system.
Team: Theori
>>> Theori's official AIxCC Landing Page: GitHub
>>> Semifinal Release: GitHub
>>> Final Release: GitHub



Buttercup

Trail of Bits' cyber reasoning system.
Team: Trail of Bits
>>> Semifinal Release: GitHub
>>> Final Release: GitHub



Fuzzing Brain

All You Need IS A Fuzzing Brain's cyber reasoning system.
Team: All You Need IS A Fuzzing Brain
>>> Semifinal Release: GitHub
>>> Final Release: GitHub



Bug Buster

42 b3yond 6ug's cyber reasoning system.
Team: 42 b3yond 6ug
>>> Semifinal Release: GitHub
>>> Final Release: GitHub



Lacrosse

Lacrosse's cyber reasoning system.
Team: Lacrosse
>>> Semifinal Release: GitHub
>>> Final Release: GitHub

CONGRATULATIONS TO TEAM



Atlanta

1st PLACE

AIxCC AI CYBER CHALLENGE → $4,000,000

DARPA + ARPA-H

COMPETITION AGGREGATE RESULTS - REAL WORLD, NON-SYNTHETIC V

ACHIEVEMENT
Ant-Man
CP: Sqlite3
First Discovered



Shrinking can lead to meeting some bugs

TEAM NAME: Team Atlanta

Semifinal

Found in C

1

Found in Java

0

Final

Found in C

6 (1 replay - SystemD)

Found in Java

12

Patched in

0

Patched in Java

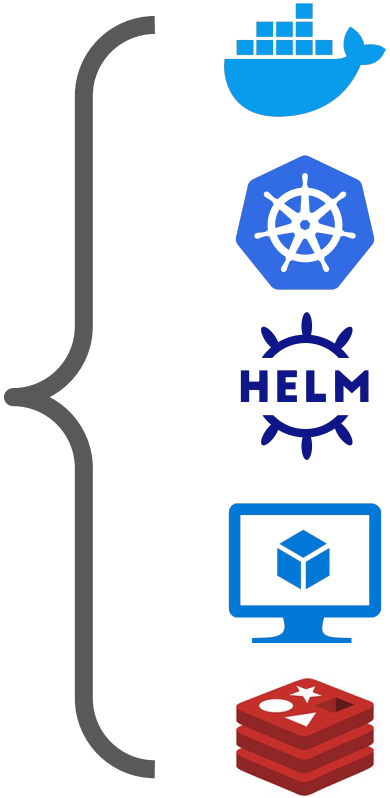
11 (3 w/o PoV)

* More information pending disclosure completion

Running All Finalist CRSs

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



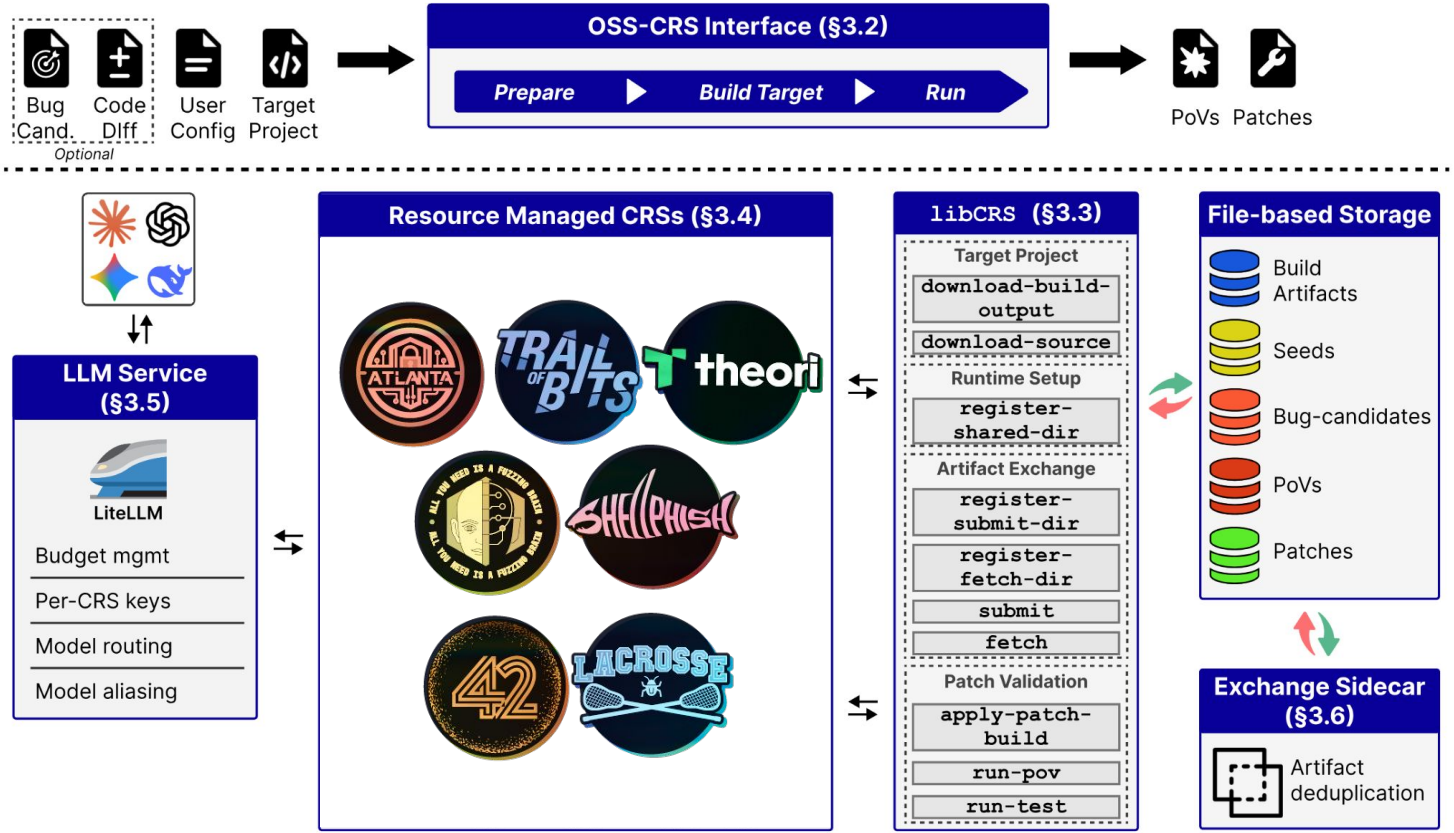


OSS-CRS: Open CRS Standard & Framework

OPEN SOURCE SecurityCon NORTH AMERICA

Georgia Tech

OpenSSF
OPEN SOURCE SECURITY FOUNDATION



What is a Cyber Reasoning System?

A **Cyber Reasoning System (CRS)** is an **evidence-producing autonomous system** that participates in an end-to-end security operation.

In the setting of the AI Cyber Challenge, a CRS **finds** and **fixes** vulnerabilities.

Given an OSS-Fuzz project, which contains fuzzing harnesses (program entrypoints)

- **Explore**: navigates the codebase to identify weaknesses.
- **Reproduce**: create an executable proof-of-vulnerability.
- **Diagnose**: triage the root cause.
- **Remediate**: proposes a patch that mitigates the proof-of-vulnerability.
- Techniques span fuzzing, static analysis, agents, or any combination.

Core Security Artifacts

The four fundamental currency primitives powering autonomous cyber reasoning systems

ENTRYPOINT & TARGET WRAPPER

Harness

Wraps target library APIs into an executable entrypoint (e.g., LLVMFuzzerTestOneInput) to ingest inputs.

CORPUS & EXPLORATION

Seed

Synthesizes well-formed test inputs that maximize code coverage and unlock deep program paths.

FAULT WITNESS & TRIGGER

Proof of Vulnerability (PoV)

Delivers an explicit input payload that reliably triggers an oracle (e.g. AddressSanitizer) crash.

DETERMINISTIC REMEDIATION

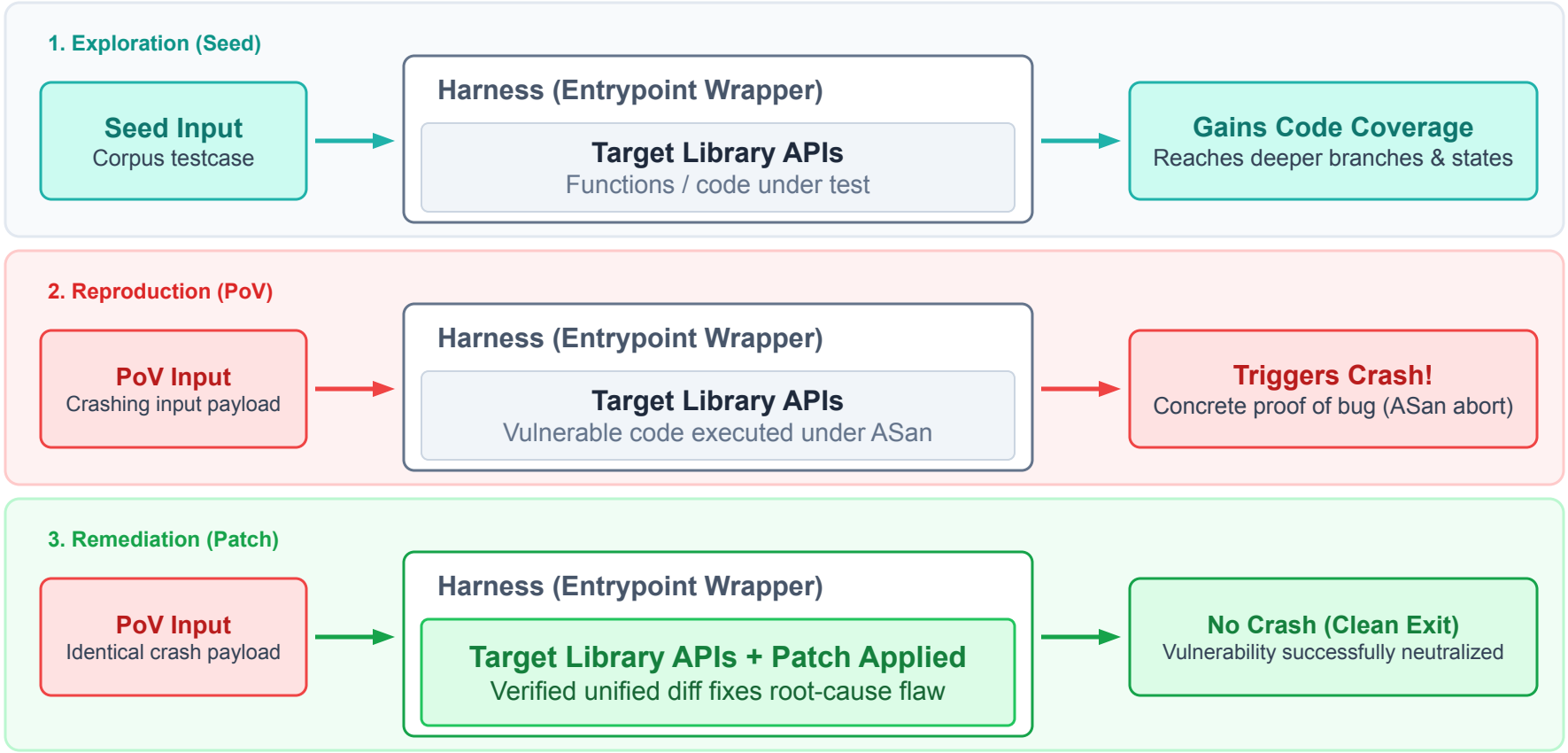
Patch

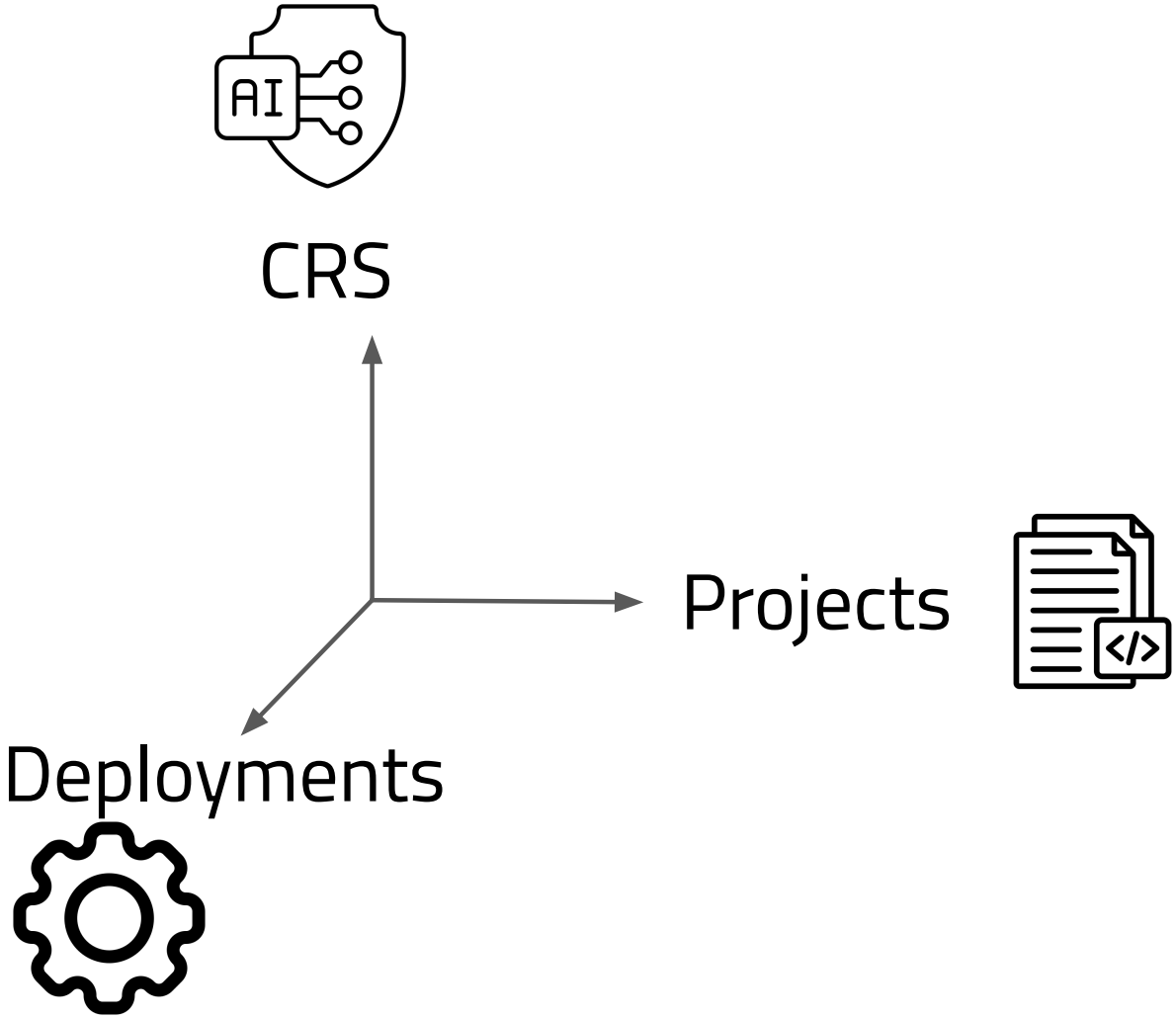
Produces a verified unified diff that eliminates the PoV crash while preserving existing regression test suites.

Core Security Artifacts

Concrete primitives powering autonomous discovery, reproduction, and remediation

OPEN SOURCE SecurityCon NORTH AMERICA




AI
CRS
Projects
Deployments

Architectural Scalability

Decoupled target, reasoning, and execution layers for enterprise open-source security

OPEN SOURCE SecurityCon NORTH AMERICA

Target Layer

Projects

Standardized Benchmarking

- Reuses Google's OSS-Fuzz standard without schema changes.
- Exposes 1,000+ real-world open-source repositories.
- Instant coverage for C, C++, and Java targets.

1,000+ Codebases

Native OSS-Fuzz Compatibility

Reasoning Layer

CRSs

Modular Cyber Reasoning Systems

- Registered via declarative, lightweight YAML manifests.
- Supports modular bug-finders, triage, and patch agents.
- Enables multi-tool ensembles and comparative evaluation.

YAML Manifest Driven

Zero-Code Module Registration

Execution Layer

Deployments

Uniform Operational Contract

- Identical execution contract from laptop to cloud.
- Runs on Docker Compose, CI/CD, or Kubernetes.
- Enables air-gapped security enclaves without code edits.

Single Contract

Laptop to Enterprise Cluster

Execution Pipeline Architecture

Three-Stage Unified Execution Lifecycle

OPEN SOURCE SecurityCon NORTH AMERICA

01 Phase 1: Prepare

Offline Environment Hydration

Builds CRS base images offline and pre-caches LLM tools, compilers, and wheels to eliminate rebuilds downstream

02 Phase 2: Build-Target

Target Compilation & Sanitizers

Compiles SUT with CRS instrumentation.

Could include instrumenting with ASAN and generating static call graphs

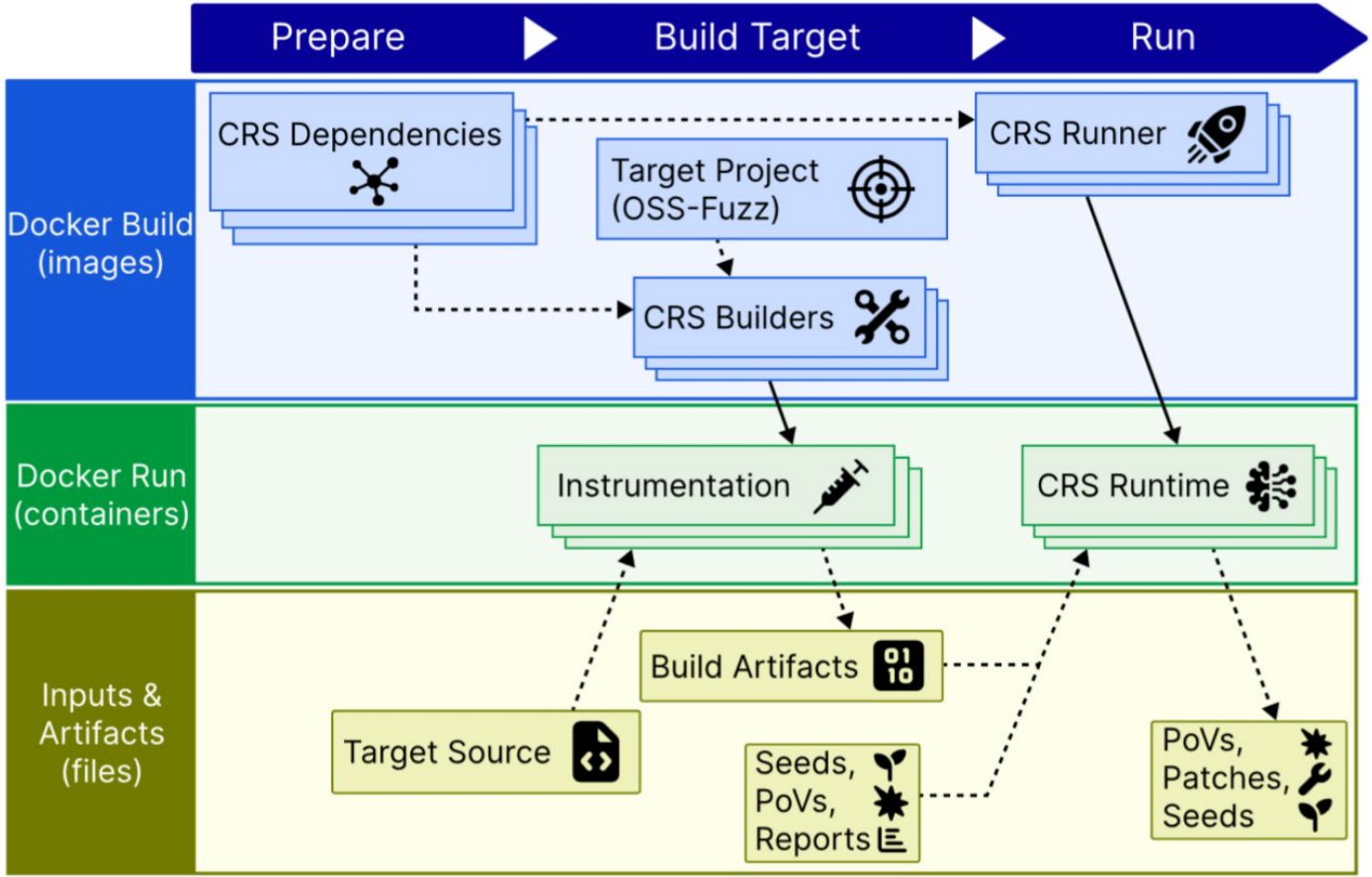
03 Phase 3: Run

Isolated Campaign Execution

Launches flat runner containers with dedicated CPU cores and mounts build artifacts

Architectural Guarantee: This flat container structure guarantees reproducible caching across projects and campaigns.

Execution Pipeline Diagram



Compute Sandboxing & Proxy Controls

Architectural isolation and governance controls for secure LLM execution

OPEN SOURCE SecurityCon NORTH AMERICA

COMPUTE SANDBOXING

Hardware Isolation & Resource Guarantees

Strict Core Pinning (cpuset)

Pins dedicated CPU cores per CRS

Memory Limits per Container

Enforces hard RAM ceilings

Separate Allocation for Helper Services

OSS-CRS infrastructure services will not be contending for resources with deployed CRSs

PROXY CONTROLS

Unified LLM Proxy Architecture

Model Aliasing (LiteLLM)

Redirects legacy models to self-hosted vLLM or frontier models without code changes.

Cost & Rate Governance

Enforces strict spending caps and tracks token consumption per target repository.

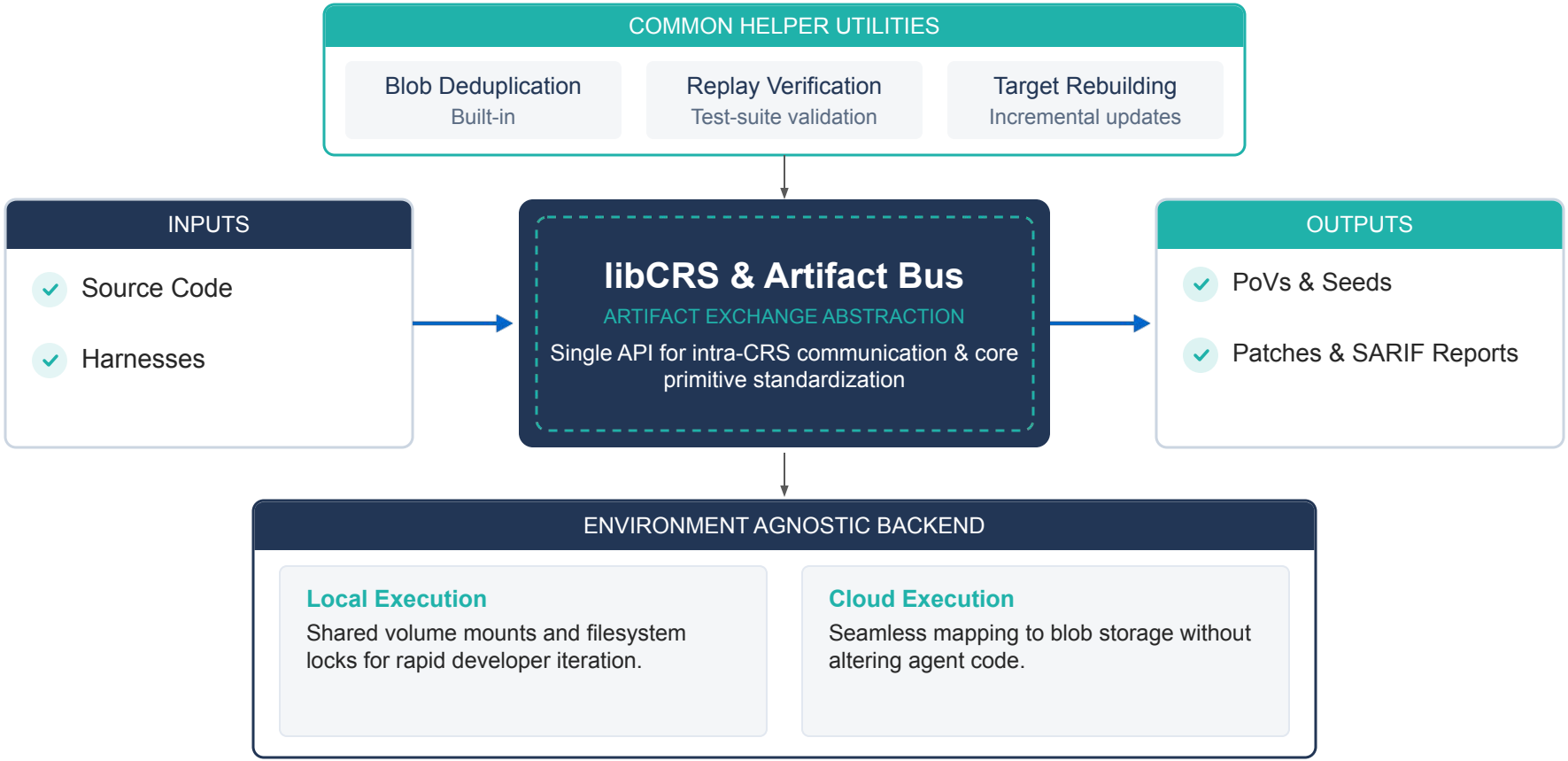
Enterprise Privacy & BYOK

Centralized API key management with local base-URL redirection for secure environments.

libCRS & Artifact Bus Architecture

Standardizing Intra-CRS Communication & Execution Workflows

OPEN SOURCE SecurityCon NORTH AMERICA



Composing CRSs

OPEN SOURCE SecurityCon NORTH AMERICA

Seed sharing

one generator feeds every fuzzer



Patch judge

many fixers, one validated patch



Parallel execution

concurrent and isolated



Modularization of AIxCC Finalist Systems

AIxCC Finalist System	Original Architecture	OSS-CRS Modularization	Role in OSS-CRS
Team Atlanta Atlantis	3 Monolithic Subsystems	Decomposed into 5 independent modules	Multilang seed generation, fuzzing, and patching
Shellphish Artiphishell	Tightly coupled CRS stack	Split into 7 swappable modules (incl. AIJON feedback & Grammaroomba)	Flexible feedback-driven fuzzing and grammar analysis
Buttercup & FuzzingBrain	Cloud fuzzing stacks	Extracted into standalone, composable tools	Dedicated corpus seed generation
Modern Coding Agents	General-purpose LLM CLIs	Integrated directly into automated pipelines	Automated patching pipelines (plug-and-play ecosystem)

Decoupled Tool Execution via libCRS

CROSS-POLLINATION PIPELINE ARCHITECTURE

OPEN SOURCE SecurityCon NORTH AMERICA

STEP 01 — EXPLORE

LLM Grammar Agent

Reverses input grammar to generate structured seed files for deep parser reach.

- Input Grammar Reversal
- Seed Corpus Generation

STEP 02 — FUZZ

Fuzzer Engine

Saturates coverage with seeded corpus to discover deep sanitizer crashes.

- Branch Saturation
- Crash Discovery (ASan)

STEP 03 — TRIAGE

Crash Triage

Captures execution traces, deduplicates PoV findings

- Sanitizer Tracing

STEP 04 — REMEDIATE

Claude-Code & Atlantis

Agents synthesize candidate fixes; LLM judge validates and merges optimal patch.

- Parallel Patching
- LLM Judge Verification

libCRS Data Bus

Artifact Pipelining Layer
Zero Direct Coupling

SEED CORPUS
Structured Inputs

→

CRASH TRACES
Heap Overflow Data

→

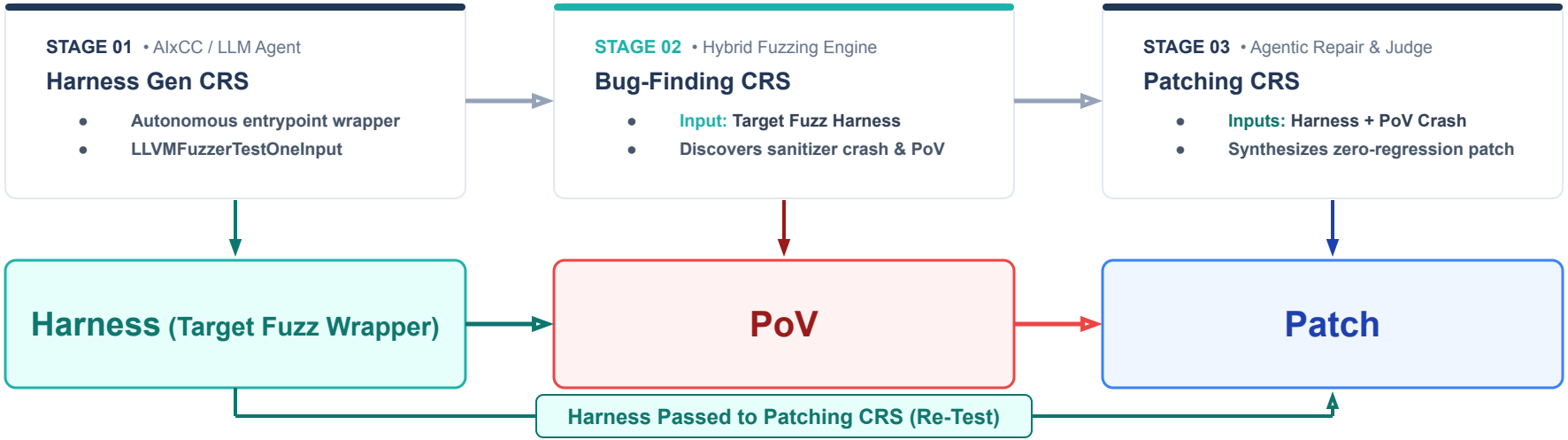
PATCH
Verified Fix

→

Autonomous Multi-CRS Coordination

ARTIFACT PIPELINING: HARNESS, POV, PATCH → CONSOLIDATED REPORT

OPEN SOURCE SecurityCon NORTH AMERICA



CONSOLIDATED DELIVERABLE: Developer Bug Report & Verified Pull Request

1. Repro Harness
Autonomous entrypoint wrapper enables instant local verification

2. PoV Crash Witness
Minimal testcase & sanitizer trace proving memory safety impact

3. Verified Patch
Zero-regression code diff ready for 1-click maintainer merge

wolfSSL: Autonomous Harness Generation

CRS created an LLVMFuzzerTestOneInput harness targeting wolfSSL's wc_PKCS7_Verify API

OPEN SOURCE SecurityCon NORTH AMERICA

TARGET INTERFACE & GAP

wc_PKCS7_Verify API

- Target API: wc_PKCS7_Verify in wolfSSL
- Coverage Gap: Unreached by standard fuzzing due to strict ASN.1 format expectations

Status: No native fuzz harness existed for this API

```
msgBuf = (byte *)malloc(msgSz);
if (msgBuf == NULL)
    return 0;
memcpy(msgBuf, data + 1, msgSz);

outBuf = (byte *)malloc(OUTPUT_BUF_SZ);
if (outBuf == NULL) {
    free(msgBuf);
    return 0;
}

pkcs7 = wc_PKCS7_New(NULL, INVALID_DEVID);
if (pkcs7 == NULL)
    goto done;

ret = wc_PKCS7_InitWithCert(pkcs7, NULL, 0);
if (ret != 0)
    goto done;

wc_PKCS7_AllowDegenerate(pkcs7, 1);
(void)wc_PKCS7_VerifySignedData(pkcs7, msgBuf, (word32)msgSz);
```

Result: New entrypoint LLVMFuzzerTestOneInput(data, size) unlocked previously unreachable cryptographic code paths.

wolfSSL: Bug-Finding & PoV Discovery

Agentic seed generation and automated AddressSanitizer crash reproduction

OPEN SOURCE SecurityCon NORTH AMERICA

VULNERABILITY MECHANISM

Heap Buffer Overflow in pkcs7.c

- **Root Cause:** Length parsing flaw in GetASN_Items during certificate decode
- **Vulnerability Class:** Heap-buffer-overflow (CWE-122) via 32-bit integer wrap-around
- **Consequence:** Malformed sequence header allowed out-of-bounds heap memory access

```
=================================================================
==3644038==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x7b2d002200ba
READ of size 1 at 0x7b2d002200ba thread T0
    #0 0x55bb7244f15d in GetASN_Items /src/wolfssl/wolfcrypt/src/asn.c:1808:32
    #1 0x55bb724647a5 in DecodeCertInternal /src/wolfssl/wolfcrypt/src/asn.c:22115
    #2 0x55bb72471bd9 in DecodeCert /src/wolfssl/wolfcrypt/src/asn.c:22415:12
    #3 0x55bb72471bd9 in ParseCertRelative /src/wolfssl/wolfcrypt/src/asn.c:23839:
    #4 0x55bb724717ef in ParseCert /src/wolfssl/wolfcrypt/src/asn.c:22788:11
    #5 0x55bb723a007a in wc_PKCS7_InitWithCert /src/wolfssl/wolfcrypt/src/pkcs7.c:
    #6 0x55bb723ab855 in PKCS7_VerifySignedData /src/wolfssl/wolfcrypt/src/pkcs7.c
    #7 0x55bb723aeee3 in wc_PKCS7_VerifySignedData /src/wolfssl/wolfcrypt/src/pkcs
    #8 0x55bb7239f7d9 in LLVMFuzzerTestOneInput /tmp/pkcs7_decode.c:66:15
    #9 0x55bb7223ce2d in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, uns
p:619:13
    #10 0x55bb72227ba2 in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigne
329:6
    #11 0x55bb7222da70 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned cha
/FuzzerDriver.cpp:865:9
    #12 0x55bb722595a2 in main /src/llvm-project/compiler-rt/lib/fuzzer/FuzzerMain
    #13 0x7f0d0122a1c9 in __libc_start_call_main csu/../sysdeps/nptl/libc_start_ca
    #14 0x7f0d0122a28a in __libc_start_main csu/../csu/libc-start.c:360:3
```

```
00000000: 0030 7406 092a 8648 86f7 0d01 0702 a035  .0t..*.H.......5
00000010: 3033 0201 0131 0f30 0d06 0960 8648 0165  03...1.0...`.H.e
00000020: 0304 0201 0500 300f 0609 2a86 4886 f70d  ......0...*.H...
00000030: 0107 01a0 0204 00a0 0a30 847f ffff fe30  .........0.....0
00000040: 64a0 0331 0000 0000 0000 0000 0000 0000  d..1............
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 00                        .......
```

Input blob to harness that triggers Address Sanitizer means we found a memory vulnerability

wolfSSL: Deterministic Patch & Verification

Autonomous patch synthesis with full regression test suite pass

OPEN SOURCE SecurityCon NORTH AMERICA

SURGICAL REMEDIATION

Arithmetic Guard in pkcs7.c

- **Bounds Enforcement:** Added defensive bounds checks on certSz before arithmetic indexing
- **Reordered Checks:** Reordered sequence length validation to reject malformed headers early
- **PoV Neutralized:** Replaying PoV against patched target no longer triggers ASAN crash

File: **wolfssl/src/pkcs7.c (clean 6-line unified diff)**

```
diff --git a/wolfcrypt/src/pkcs7.c b/wolfcrypt/src/pkcs7.c
index 882a2af..8c41f62 100644
--- a/wolfcrypt/src/pkcs7.c
+++ b/wolfcrypt/src/pkcs7.c
@@ -7756,11 +7756,13 @@ static int PKCS7_VerifySignedData(wc_PKCS7* pkcs7, con
                             ret = ASN_PARSE_E;

                         cert = &pkiMsg2[idx];
-                        certSz += (int)(certIdx - idx);
-                        if (certSz > length) {
+                        /* Check before adding header size to avoid integer
+                         * overflow of certSz. */
+                        if (certSz > length - (int)(certIdx - idx)) {
                             ret = BUFFER_E;
                             break;
                         }
+                        certSz += (int)(certIdx - idx);
                     }
         #ifdef ASN_BER_TO_DER
                     der = pkcs7->der;
```

Autonomous Lifecycle Complete: Harness Generation → PoV Reproduction → Patch Synthesis → Oracle Verification.

Roadmap Tracks

Strategic directions across defense security, academic programs, and workflow automation

OPEN SOURCE SecurityCon NORTH AMERICA

DEFENSE TRACK

Institutional Partnerships

Collaborating with MIT Lincoln Lab & DARPA.

Air-Gapped Infrastructure

Offline local models for classified enclaves.

EDUCATION TRACK

Community Competitions

Powering Mini-AIxCC challenges with NYU & CSAW.

Academic Benchmarking

Lowering hardware barriers for student research.

AUTOMATION TRACK

Workflow Integration

Automating triage and PR patching in GitHub Actions.

Get Involved & Join the Community

OPEN SOURCE SecurityCon NORTH AMERICA

GET INVOLVED — COMMUNITY RESOURCES

CODE

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

COMMUNITY SLACK CHANNEL

https://openssf.slack.com/archives/C0BAY36H2DB

BENCHMARK SUITE

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

Open community participation, code contributions, and modular CRS adoption are welcome.

https://oss-crs.openssf.org

SANDBOX OpenSSF