OSS-CRS: Liberating AIxCC Cyber Reasoning Systems for Open-Source Security

Andrew Chin, Dongkwan Kim, Yu-Fu Fu, Fabian Fleischer, Youngjoon Kim, HyungSeok Han, Cen Zhang, Brian Junekyu Lee, Hanqing Zhao, and Taesoo Kim

Georgia Institute of Technology and Microsoft

Georgia Tech

Agentic Security is Here





Georgia Tech

DARPA + ARPA-H



AIxCC
AI CYBER CHALLENGE

WHAT IS AIxCC?

- A competition that rewards autonomous systems that find and patch vulnerabilities in source code.
- 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.

FINAL ROUND DATA POINTS

Total Known Vulnerabilities
70

Real World Vulns discovered
18

Total spent (Compute + LLM)
$359k

Vulnerabilities discovered
54 (77%)

Average time to patch
45 min

Total LLM queries
1.9M

Vulnerabilities patched
43 (61%)

Total LOC analyzed
54M

LLM Spend
$82k

AIxCC
AI CYBER CHALLENGE

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.

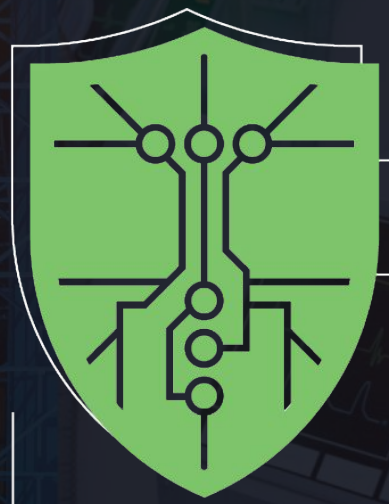
In the setting of the AI Cyber Challenge, a CRS **finds** and **fixes** vulnerabilities.

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.



Preliminary events



Top 7 teams advance



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

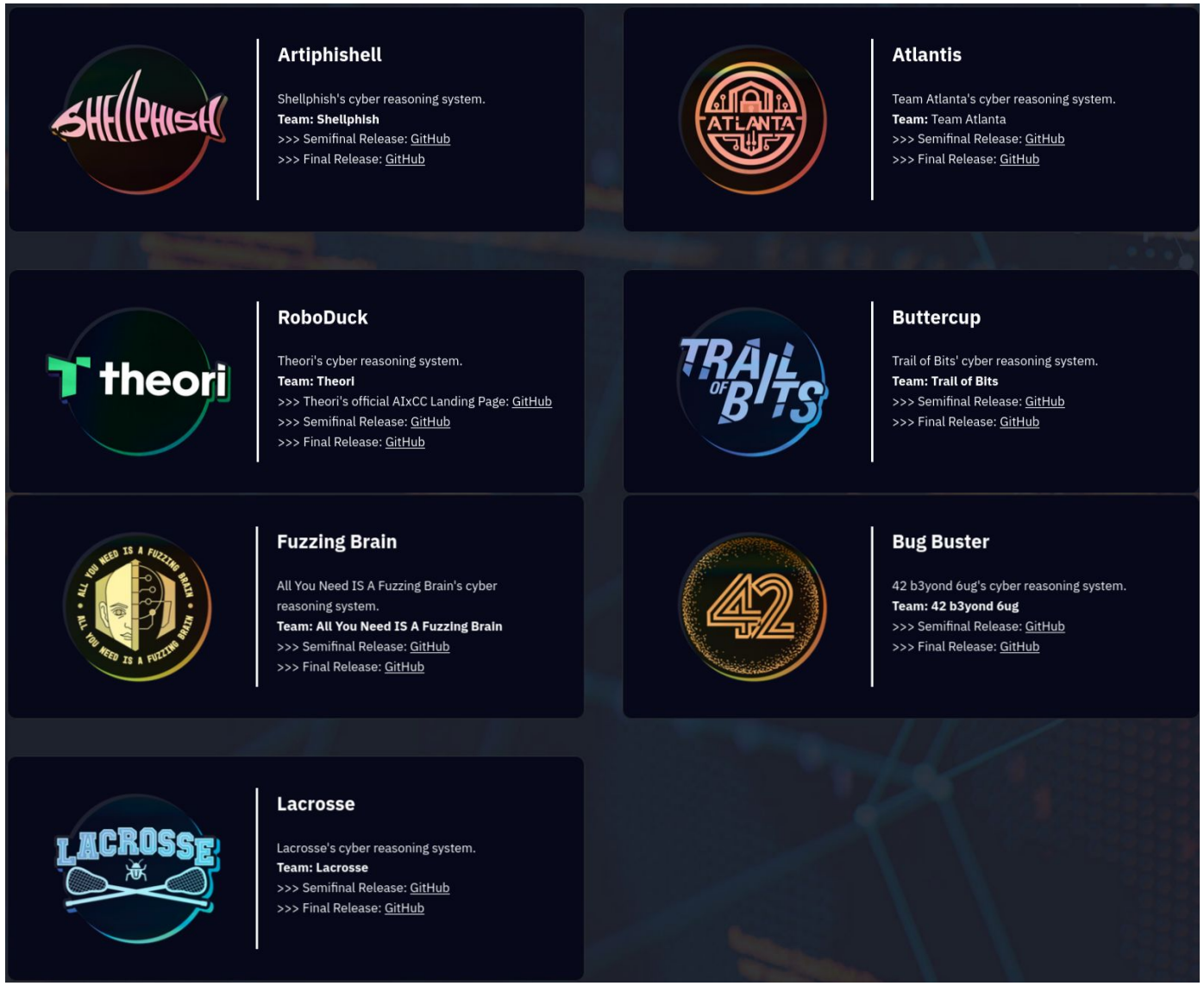
1ST: $4 MILLION

2ND: $3 MILLION

3RD: $1.5 MILLION

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

Open Sourced!



Georgia Tech

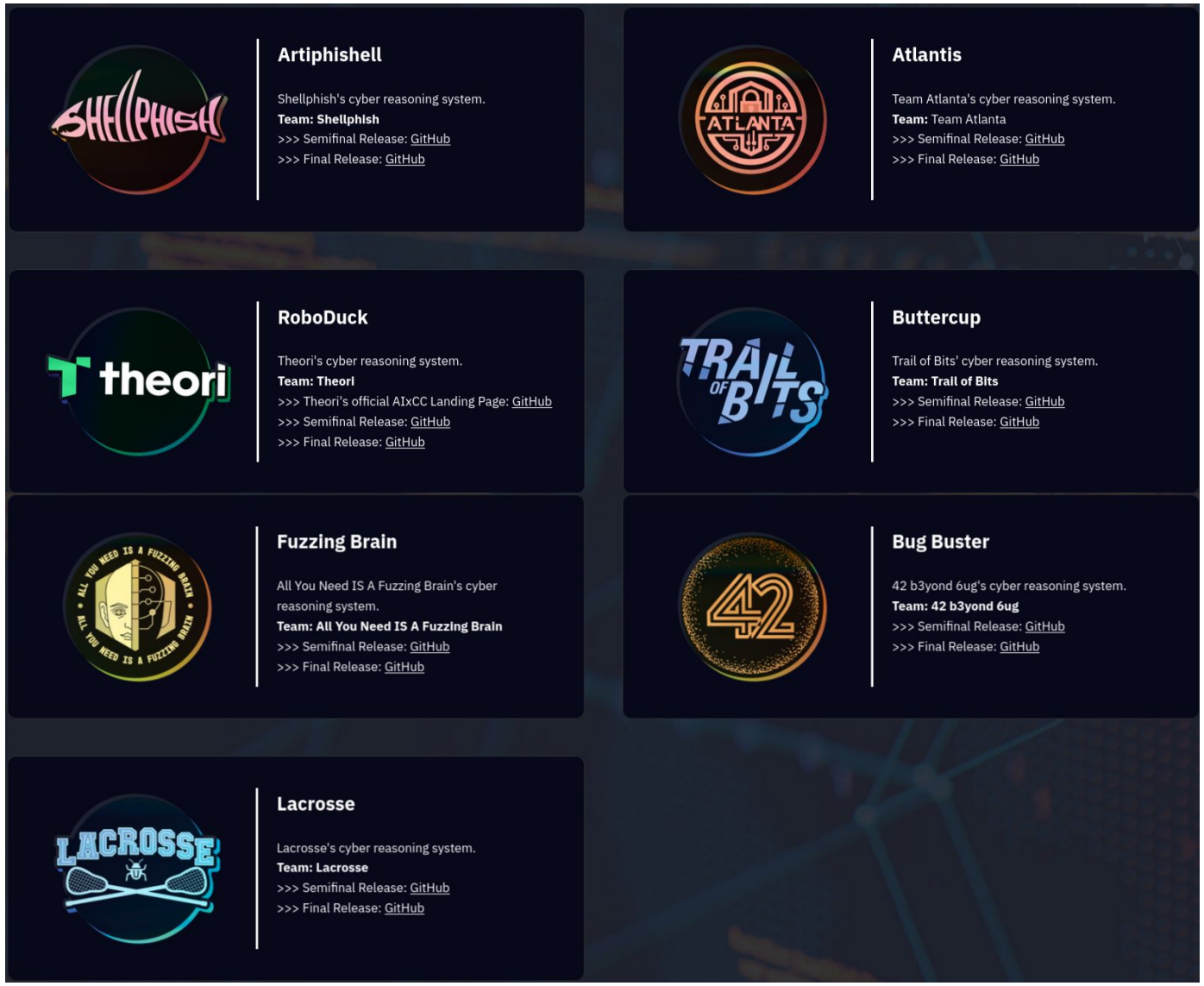
Open Sourced! But Largely Unmaintained…

No Activity

Archived

Active

No Activity



Archived

Active

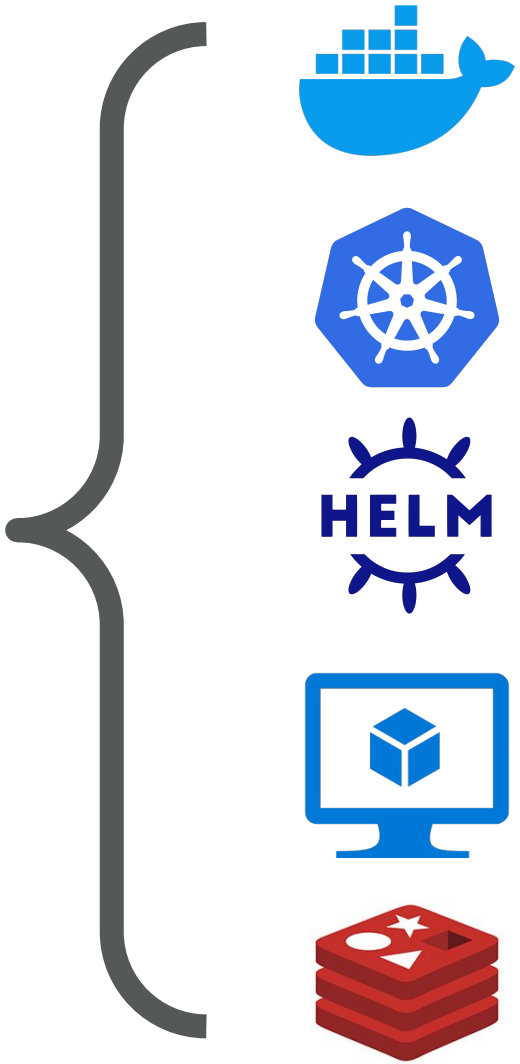
No Activity

Georgia Tech

Challenges of Running All Finalist CRSs

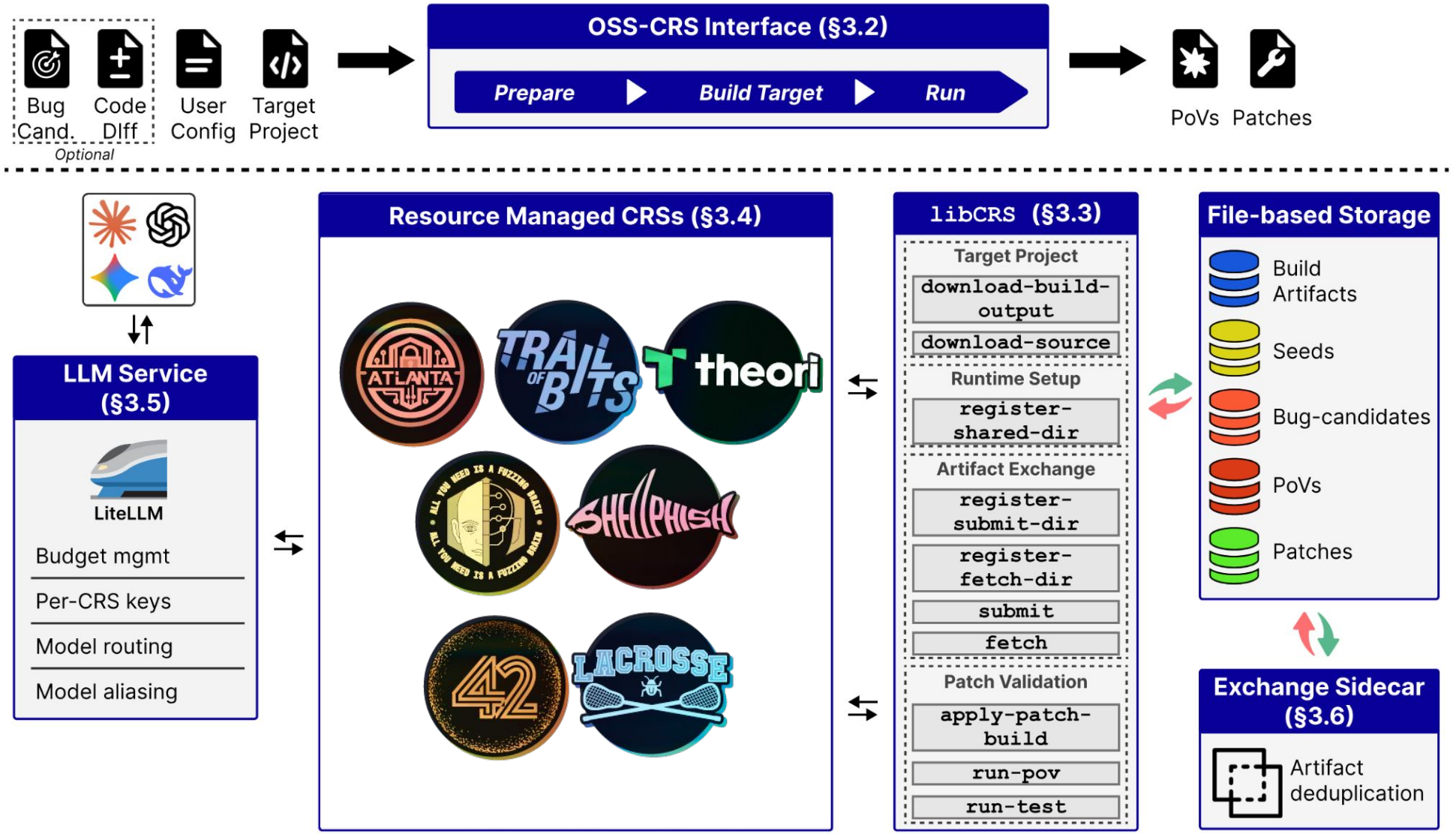
- ✗ Cloud Lock-In
- ✗ Infrastructure Duplication
- ✗ Monolithic Design



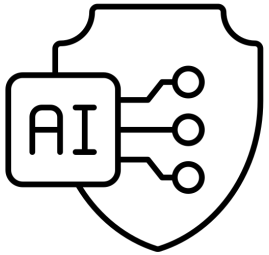


Georgia Tech

OSS-CRS: Open CRS Standard & Framework



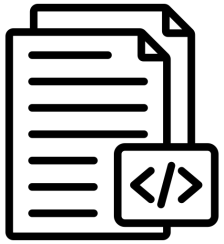
Georgia Tech



CRS

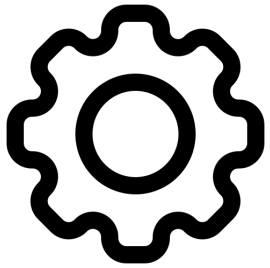
Scales with Cyber Reasoning Systems
registered in OSS-CRS
(e.g. bug-finding, patching)

Projects



Scales with Open-Source Software
compatible with Google's OSS-Fuzz
(i.e. Dockerfile, build.sh, run_tests.sh)

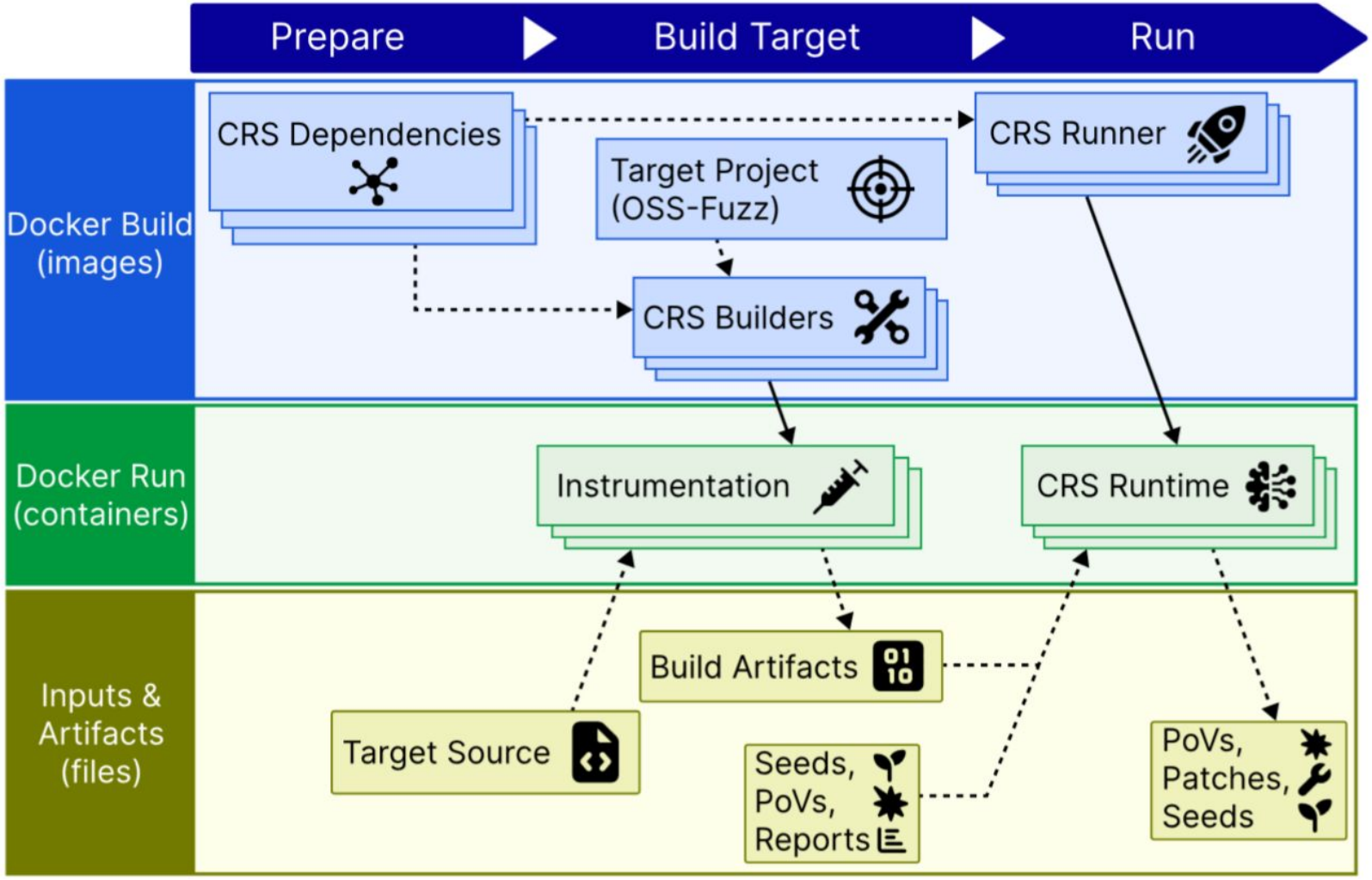
Deployments



Scales with features
supported in OSS-CRS
(e.g. resource management)

Georgia Tech

Local Deployment



Georgia Tech

LLM Controls



Budget Management

Each CRS can be declared to run with maximum budget

Secret Forwarding

Alternatively, auth secrets may be forwarded directly to CRSs

Model Aliasing

CRS may request to hard-coded model names, which are forwarded to the operator's desired models defined in configuration

Base URL and BYOK

Direct requests to any OpenAI-compatible endpoint, such as vLLM or LiteLLM

Georgia Tech

Composing CRSs

Seed sharing

one generator feeds every fuzzer



Patch judge

many fixers, one validated patch



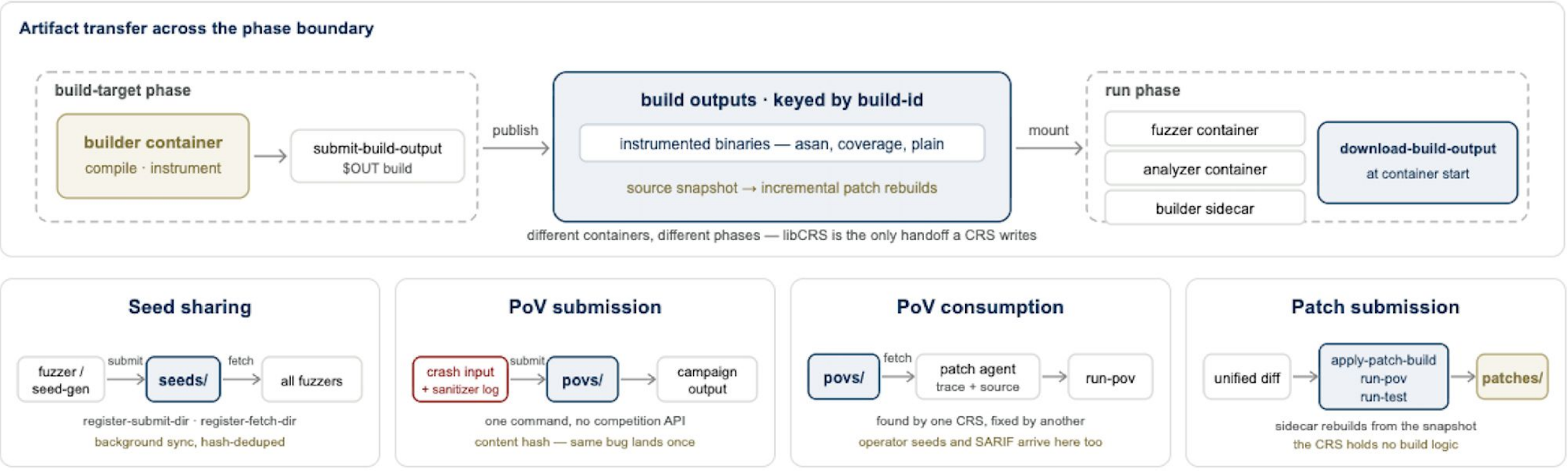
Parallel execution

concurrent and isolated



Georgia Tech

libCRS: CRS-Facing Interface



Georgia Tech

Bugs Found Using OSS-CRS

Deployed Atlantis-Multilang and Atlantis-C*

7
Vulns

4
Via Existing Harnesses

<$50
Per bug found

PROJECT	VULNERABILITY	CVSS	REFERENCE
libxml2 C · 130k	OOM Error Misreport CWE-252	2.5	#1067
libyang C · 115k	Infinite Recursion CWE-674	7.5	GHSA-mq87-gwqp-w3v8
FRRouting C · 627k	NULL Pointer Deref CWE-476	6.5	PR #20932
memcached C · 32k	Heap Buffer Underflow CWE-125	3.9	#1268
jq C · 30k	Float-to-int Overflow CWE-681	3.3	#3482
jq C · 30k	Null Pointer Arithmetic CWE-476	2.5	#3483
arp-scan C · 3k	*Redacted for responsible disclosure*	5.4	—

*Atlantis-Java also deployed with real findings, but the results are not included in paper

Georgia Tech

Case Study: Atlantis-Multilang on libyang

WEIGHTED DISPATCH

```
strategies = [
    # ...
    (10, build_malformed_module),
    (10, build_pattern_heavy_module),
    (10, build_xpath_heavy_module),
    (10, build_submodule),
]
total = sum(w for w, _ in strategies)
r = rnd.randint(0, total - 1)
cumul = 0
chosen_fn = strategies[-1][1]
for w, fn in strategies:
    cumul += w
    if r < cumul:
        chosen_fn = fn
        break
try:
    payload = chosen_fn(rnd)
except Exception:
    payload = 'module fallback { ... }'
```

THE CHOSEN STRATEGY

```
def build_xpath_heavy_module(rnd):
    name   = rand_id(rnd, 3, 15)
    prefix = rand_id(rnd, 2, 8)
    lines  = [f'module {name} {{']
    lines.append(f'  yang-version 1.1;')
    lines.append(f'  namespace "urn:example:{name}";')
    lines.append(f'  prefix {prefix};')

    xpath_exprs = [
      f'count(../item[.>0]) > {rnd.randint(0, 100)}',
      f'string-length(normalize-space(../name))'
      f' <= {rnd.randint(1, 255)}',
      f'boolean(deref(../leafref-leaf)/../value)',
      f'../a = ../b or ../c != ../d',
      f'(../x + ../y) * ../z <= {rnd.randint(1, 1000)}',
      f'sum(../values/item) > 0',
      # ...
    ]

    container_name = rand_id(rnd)
    lines.append(f'  container {container_name} {{')
    for _ in range(rnd.randint(2, 8)):
        leaf_name = rand_id(rnd)
        expr = rnd.choice(xpath_exprs)
        lines.append(f'    leaf {leaf_name} {{')
        lines.append(f'      type string;')
        lines.append(f'      must "{expr}";')
        if rnd.random() < 0.3:
            when_expr = rnd.choice(xpath_exprs)
            lines.append(f'      when "{when_expr}";')
        lines.append(f'    }}')
    lines.append(f'  }}')
    lines.append('}')
    return '\n'.join(lines)
```

RESULTING SEED — 764 bytes

```
module kyv {
  yang-version 1.1;
  namespace "urn:example:kyv";
  prefix ae_FOPb;
  container efLQCbAP {
    leaf Na7GD7vNMFCjj {
      type string;
      must "boolean(deref(../leafref-leaf)/../value)";
    }
    leaf nVowBz54G8l5FH {
      type string;
      must "count(../item[.>0]) > 9";
    }
    leaf M9CfTIEDkkKKWQWK..6l {
      type string;
      must "count(../item[.>0]) > 9";
      when "../a = ../b or ../c != ../d";
    }
    leaf tlVaEzAaL5m {
      type string;
      must "../a = ../b or ../c != ../d";
    }
    leaf wQ_HaYVEzS07 {
      type string;
      must "(../x + ../y) * ../z <= 588";
      when "string-length(normalize-space(../name)) <= 44";
    }
    leaf LETAY8O {
      type string;
      must "sum(../values/item) > 0";
    }
  }
}
```

Case Study: libyang infinite recursion

PoC input

yang_module.yang

```
module leafref-unions {
  namespace "http://example.com/lu";
  prefix "lu";

  container config {
    leaf circular-ref-1 {
      type union {
        type leafref {
          // resolves back to itself
          path "../circular-ref-1";
        }
      }
      // forces the default to be stored
      default "circular-value";
    }
  }
}
```

Vulnerable code

src/plugins_types/union.c, leafref.c

```
/* union.c:495 — lyplg_type_store_union() */
ret = union_find_type(ctx, type_u, subvalue, ...);

/* union.c:346 — union_find_type() */
for (u = 0; u < LY_ARRAY_COUNT(type_u->types); ++u) {
    ret = union_store_type(ctx, type_u, u, subvalue, ...);

/* union.c:219,275 — union_store_type() */
struct lysc_type *type = type_u->types[type_idx];  // the leafref
type_plg = LYSC_GET_TYPE_PLG(type->plugin_ref);
rc = type_plg->store(ctx, type, value, ...);       // -> leafref.c:60

/* leafref.c:60 — lyplg_type_store_leafref() */
rc = LYSC_GET_TYPE_PLG(type_lr->realtype->plugin_ref)
        ->store(ctx, type_lr->realtype, value, ...);
        // realtype == the same union, so ->store is ...

/* union.c:778 — the union plugin's vtable */
const struct lyplg_type_record plugins_union[] = {{
    .name         = LY_TYPE_UNION_STR,
    .plugin.store = lyplg_type_store_union,  // ... back to the top
}};
// no depth limit, no visited-type check
// => unbounded recursion, ASAN stack-overflow
```

Georgia Tech

TEAM	COMPONENT	CAPABILITY	LLM	LANGUAGE	DESCRIPTION
Vanilla fuzzer	crs-libfuzzer	FINDING	✗	C/C++	libFuzzer engine; minimal baseline CRS
	crs-jazzer	FINDING	✗	Java	Jazzer JVM fuzzer; minimal baseline CRS
Atlantis	atlantis-multilang-given_fuzzer	FINDING	✗	C/C++/Java	Base UniAFL fuzzer
	atlantis-multilang-wo-concolic	FINDING	✓	C/C++/Java	UniAFL fuzzer with LLM mutator microservices
	atlantis-c-deepgen	FINDING	✓	C/C++	LibAFL fuzzer with LLM seed generation
	atlantis-c-bullseye	FINDING	✓	C/C++	AFL++ directed fuzzer
	atlantis-java-main	FINDING	✓	Java	Jazzer sinkpoint-directed fuzzing
	crs-prism	FIXING	✓	C/C++/Java	Cyclic analyze–patch–evaluate agent
	crs-vincent	FIXING	✓	C/C++/Java	Root-cause and property-analysis patch agent
	crs-multi-retrieval	FIXING	✓	C/C++/Java	Two-step retrieval-based patch agent
BugBuster	42-directed	FINDING	✗	C/C++	AFL++ directed fuzzer
	42-seedgen	FINDING	✓	C/C++/Java	LLM seed-generation agent
Buttercup	buttercup-seed-gen	FINDING	✓	C/C++/Java	LLM seed-generation agent
Artiphishell	crs-shellphish-c-fuzzers-libfuzzer	FINDING	✗	C/C++	libFuzzer fuzzers
	crs-shellphish-c-fuzzers-aflpp	FINDING	✗	C/C++	AFL++ fuzzers
	crs-shellphish-jvm-fuzzers	FINDING	✗	Java	Jazzer fuzzer with LOSAN sanitizer
	crs-shellphish-aijon	FINDING	✓	C/C++	LLM IJON annotation generation and fuzzing
	crs-shellphish-discoveryguy	FINDING	✓	C/C++	LLM vulnerability discovery
	crs-shellphish-grammar	FINDING	✓	C/C++/Java	Grammar-based fuzzer
	crs-shellphish-quickseed	FINDING	✓	C/C++/Java	LLM seed-generation agent
RoboDuck	roboduck	FINDING	✓	C/C++/Java	LLM seed- and PoV-generation agent
FuzzingBrain	fuzzing-brain	FINDING	✓	C/C++/Java	LLM PoV-generation agent
Lacrosse	lacrosse-seed-gen	FINDING	✓	C/C++/Java	LLM seed-generation agent
AI coding agent	crs-bug-finding-claude-code	FINDING	✓	C/C++/Java	Claude Code coding agent for bug finding
	crs-bug-finding-codex	FINDING	✓	C/C++/Java	Codex CLI coding agent for bug finding
	crs-bug-finding-copilot-cli	FINDING	✓	C/C++/Java	Copilot CLI coding agent for bug finding
	crs-bug-finding-gemini-cli	FINDING	✓	C/C++/Java	Gemini CLI coding agent for bug finding
	crs-bug-finding-opencode	FINDING	✓	C/C++/Java	OpenCode coding agent for bug finding
	crs-claude-code	FIXING	✓	C/C++/Java	Claude Code coding agent for patch generation
	crs-codex	FIXING	✓	C/C++/Java	Codex CLI coding agent for patch generation
	crs-copilot-cli	FIXING	✓	C/C++/Java	Copilot CLI coding agent for patch generation
	crs-gemini-cli	FIXING	✓	C/C++/Java	Gemini CLI coding agent for patch generation
	crs-opencode	FIXING	✓	C/C++/Java	OpenCode coding agent for patch generation

Get Involved

Community actively working on

- New Cyber Reasoning Systems
- Deployment features (e.g. CI/CD, offline mode)
- Expanded CRS scope and capabilities (e.g. harness generation)



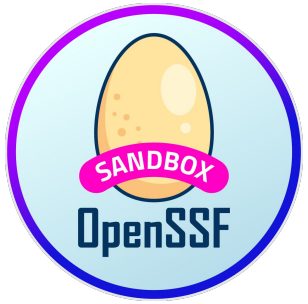




https://github.com/ossf/oss-crs

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





OpenSSF and the OpenSSF logo design are trademarks of The Linux Foundation. See openssf.org for more information.
This presentation was created using resources from Flaticon.com

Georgia Tech