The journey from AIxCC to Samsung Internal AI-powered Security Solution

장 준 언

Samsung Electronic (DX) / AI Platform Center / Security & Privacy Team

보안 점검 자동화 기술 연구 개발: Fuzzing / Static Analysis / AI Agent

linkedin.com/joonun-jang

AIxCC: DARPA AI Cyber Challenge

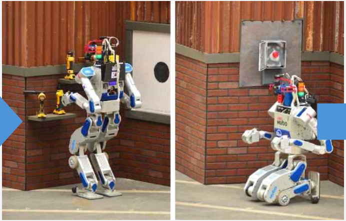
DARPA + ARPA H

AIxCC
AI CYBER CHALLENGE

PATCHING CRITICAL INFRASTRUCTURE

DARPA's AI Cyber Challenge
Announced on 08/2023

2004-2007
(Autonomous Vehicle)

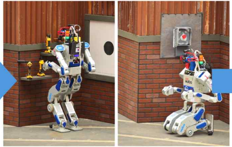


2012-2015
(Robotics Challenge)

2014-2016
(Cyber Grand Challenge)

[Ref. https://en.wikipedia.org/wiki/DARPA_Grand_Challenge]

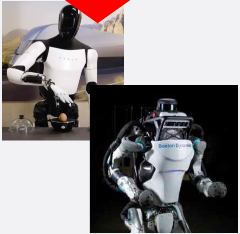
2004-2007
(Autonomous Vehicle)



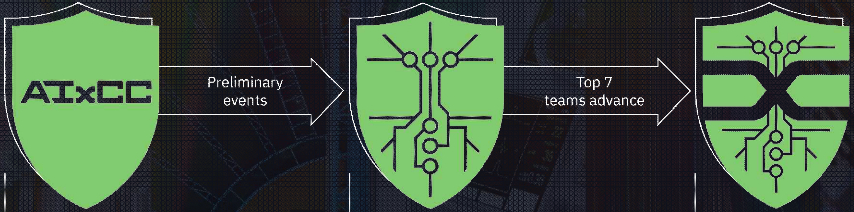
2012-2015
(Robotics Challenge)

2014-2016
(Cyber Grand Challenge)

20+ years



Mayhem
by ForAllSecure



AIxCC

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 ANTHROPIC OpenAI Microsoft THE LINUX FOUNDATION OpenSSF OPEN SOURCE SECURITY FOUNDATION

DARPA + ARPA-H

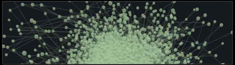
AIxCC

AI CYBER CHALLENGE

WHAT IS AIxCC?

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

Security Tasks in AIxCC

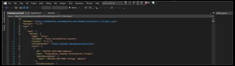


Proof-Of-Vulnerability (POV)

- Input data to reproduce vulnerability crash in harness

PATCH

- Unified diff source code fix for vulnerabilities



```
{
  "$schema": "https://schemastore.azurewebsites.net/schemas/json/sarif-2.1.0-rtm.5.json",
  "version": "2.1.0",
  "runs": [
    {
      "tool": {
        "driver": {
          "name": "Trivy",
          "fullName": "Trivy Vulnerability Scanner",
          "version": "0.17.2",
          "informationUri": "https://github.com/aquasecurity/trivy",
          "rules": [
            {
              "id": "RUSTSEC-2019-0001/ammonia",
              "name": "Programming Language Vulnerability (Cargo)",
              "shortDescription": {
                "text": "RUSTSEC-2019-0001 Package: ammonia"
              },
              "fullDescription": {
```

SARIF Assessment

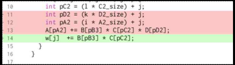
- Structured reporting format for vulnerability details

Archives: 1
Folders: 0
Files: 15
Size: 7108839 (6 MB)
Compressed size: 1290263 (1 MB)

There are no errors

BUNDLE

- Grouping of related PoV, patch, and SARIF submissions



```
            int pC2 = (l * C2_size) + j;
            int pD2 = (k * D2_size) + j;
            int pA2 = (i * A2_size) + j;
            A[pA2] += B[pB3] * C[pC2] * D[pD2];
            w[j]  += B[pB3] * C[pC2];
        }
    }
```

DELTA SCAN

- Challenge analyzing base code plus applied diff changes

FULL SCAN

- Challenge analyzing entire code base

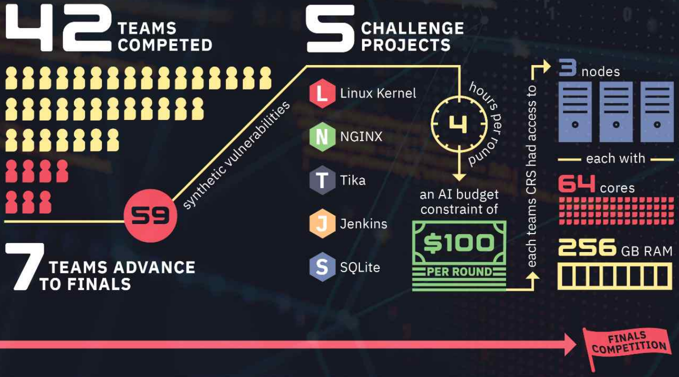
AIxCC

AI CYBER CHALLENGE

SEMIFINAL COMPETITION
OVERVIEW

DARPA + ARPA-H

To help secure our critical infrastructure, teams created custom CRSs that competed in the AIxCC Semifinal Competition.



42 TEAMS COMPETED

59 synthetic vulnerabilities

7 TEAMS ADVANCE TO FINALS

5 CHALLENGE PROJECTS

- L Linux Kernel
- N NGINX
- T Tika
- J Jenkins
- S SQLite

4 hours per round

an AI budget constraint of

$100 PER ROUND

each teams CRS had access to

3 nodes

each with

64 cores

256 GB RAM

COLLABORATORS & PARTNERS

Google · ANTHROP\C · OpenAI · Microsoft · THE LINUX FOUNDATION · OpenSSF OPEN SOURCE SECURITY FOUNDATION · black hat · DEFCON

CONGRATULATIONS TO TEAM

Atlanta

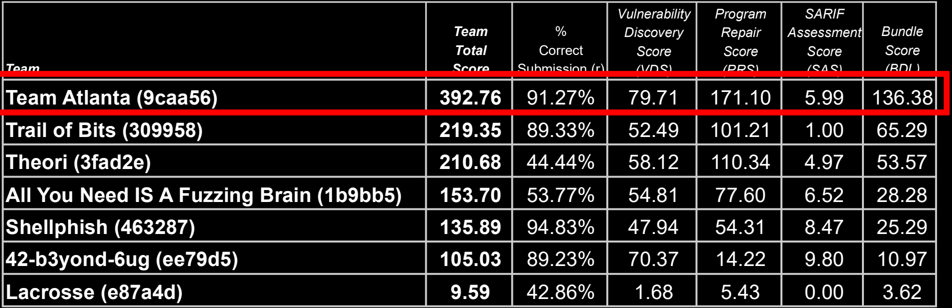
1st PLACE

AIxCC
AI CYBER CHALLENGE

→ **$4,000,000**

DARPA + ARPA-H

Scoreboard breakdown



Team	Team Total Score	% Correct Submission (r)	Vulnerability Discovery Score (VDS)	Program Repair Score (PRS)	SARIF Assessment Score (SAS)	Bundle Score (BDL)
Team Atlanta (9caa56)	**392.76**	91.27%	79.71	171.10	5.99	136.38
Trail of Bits (309958)	**219.35**	89.33%	52.49	101.21	1.00	65.29
Theori (3fad2e)	**210.68**	44.44%	58.12	110.34	4.97	53.57
All You Need IS A Fuzzing Brain (1b9bb5)	**153.70**	53.77%	54.81	77.60	6.52	28.28
Shellphish (463287)	**135.89**	94.83%	47.94	54.31	8.47	25.29
42-b3yond-6ug (ee79d5)	**105.03**	89.23%	70.37	14.22	9.80	10.97
Lacrosse (e87a4d)	**9.59**	42.86%	1.68	5.43	0.00	3.62

$$Team\ Score = \sum Challenge\ Scores$$

$$Challenge\ Score = AM * (VDS + PRS + SAS + BDL)$$

$$AM = 1 - (1 - r)^4$$

All projects we adapted into challenges

S2N-TLS

LITTLE-CMS

DICOOGLE

LIBPNG

WIRESHARK

NDPI

XZ

JSOUP

MONGOOSE

LIBPOSTAL

SQLITE

FREEDP

TIKA

HERTZBEAT

LIBAVIF

HEALTHCARE-DATA-HARMONIZE

PDFBOX

OPENSSL

SYSTEMD

SHADOWSOCKS-LIBEV

DCMCHE

LIBEXIF

ZOOKEEPER

IPF

LIBXML2

LOGGING-LOG4J2

COMMON-COMPRESS

LWIP

POI

CURL

FREERTOS-KERNEL

COMPETITION AGGREGATE RESULTS - SYNTHETIC VULNERABILITIES

Semifinal
(5 Repositories / 59 Challenges)

Vulnerabilities discovered

37% (22/59)

Vulnerabilities patched

25% (15/59)

Avg. Time to patch

2 hours

Final
(28 Repositories / 53 Challenges)

Known Vulnerabilities discovered

77% (54/70)

Known Vulnerabilities patched

61% (43/70)

Avg. Time to patch

45 minutes

COMPETITION AGGREGATE RESULTS - REAL WORLD, NON-SYNTHETIC VULNERABILITIES

Semifinal

Found in C

1

Found in Java

0

Final

Found in C

6 (1 replay - SystemD)

Found in Java

12

Patched in C

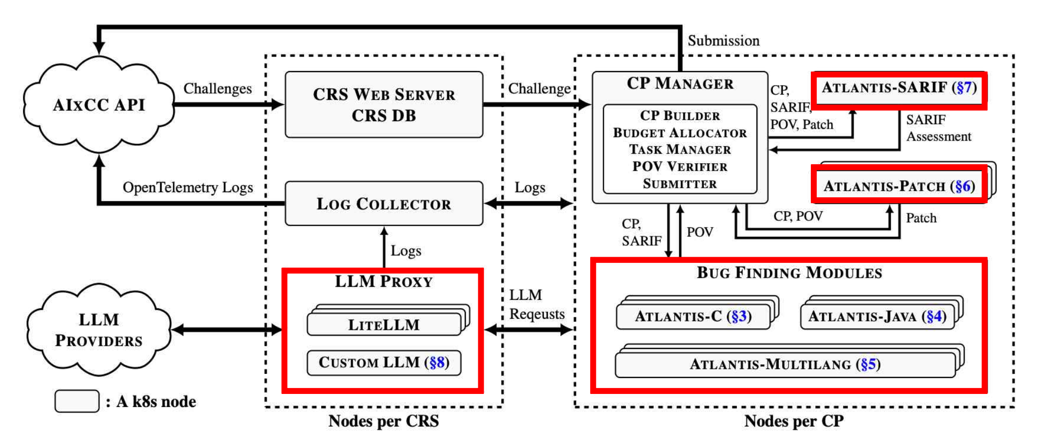
0

Patched in Java

11 (3 w/o PoV)

* More information pending disclosure completion

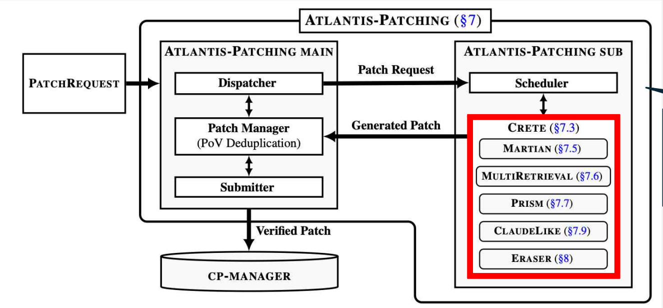
Atlantis Overview



[Ref. ATLANTIS: AI-driven Threat Localization, Analysis, and Triage Intelligence System]

Patching: Ensemble of Six Agents

- Motivation
  - *PoV as oracle, yet SLOW* → generating high quality patches
  - Six independent agents with orthogonal spectrum of decision decisions (e.g., sophisticated tool calls, no tool calls, #shots, reasoning models, etc)

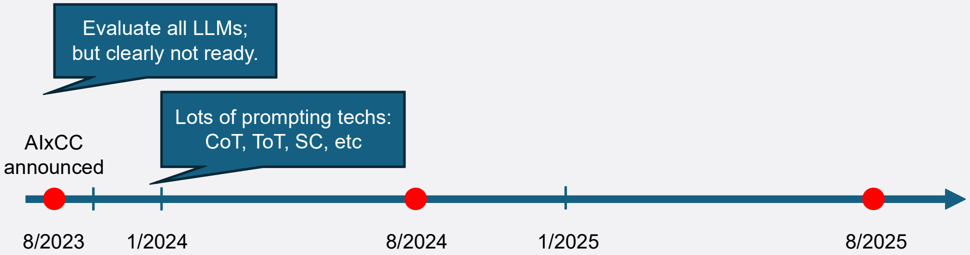


Six independent agents to generate patches

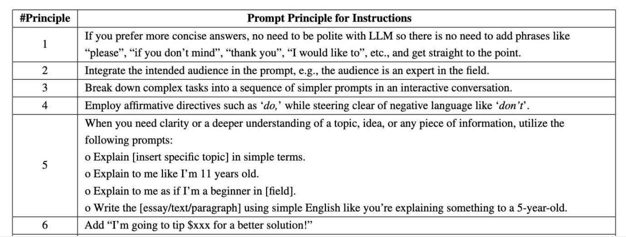
[Ref. ATLANTIS: AI-driven Threat Localization, Analysis, and Triage Intelligence System]

Our Journey

Starting as AI Skeptics (2024-)



Starting as AI Skeptics (2024-)

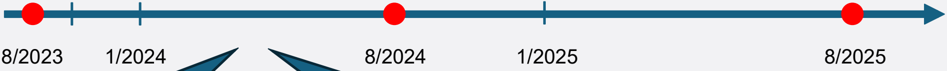


#Principle	Prompt Principle for Instructions
1	If you prefer more concise answers, no need to be polite with LLM so there is no need to add phrases like "please", "if you don't mind", "thank you", "I would like to", etc., and get straight to the point.
2	Integrate the intended audience in the prompt, e.g., the audience is an expert in the field.
3	Break down complex tasks into a sequence of simpler prompts in an interactive conversation.
4	Employ affirmative directives such as '*do,*' while steering clear of negative language like '*don't*'.
5	When you need clarity or a deeper understanding of a topic, idea, or any piece of information, utilize the following prompts: o Explain [insert specific topic] in simple terms. o Explain to me like I'm 11 years old. o Explain to me as if I'm a beginner in [field]. o Write the [essay/text/paragraph] using simple English like you're explaining something to a 5-year-old.
6	Add "I'm going to tip $xxx for a better solution!"

Evaluate all LLMs; but clearly not ready.

Lots of prompting techs: CoT, ToT, SC, etc

AIxCC announced

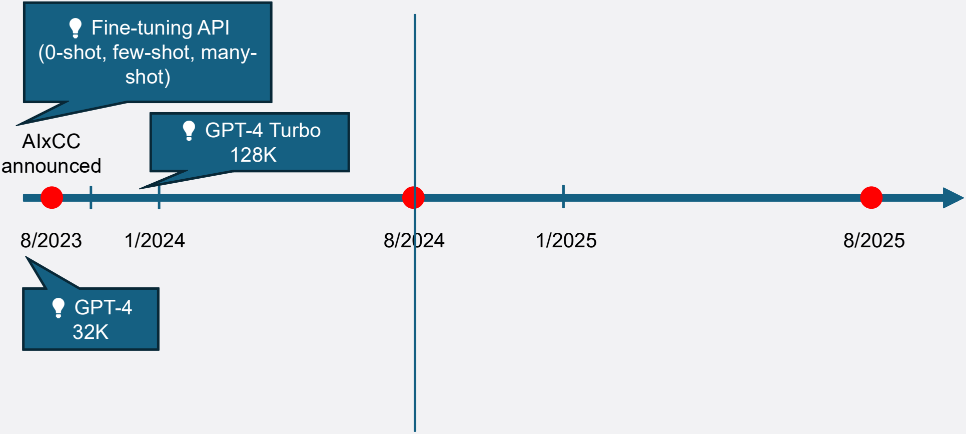


Discovering magic tricks in prompts: gaslighting, tipping, affirmative, etc

Primitive discussions: e.g., udiff gen? search-replace? full function gen?

[Ref. Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4, 2024/01]

Context Window is Fundamental Prob?



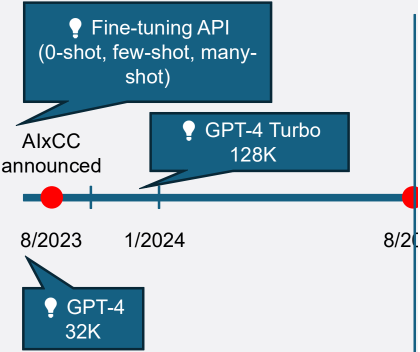
Fine-tuning API (0-shot, few-shot, many-shot)

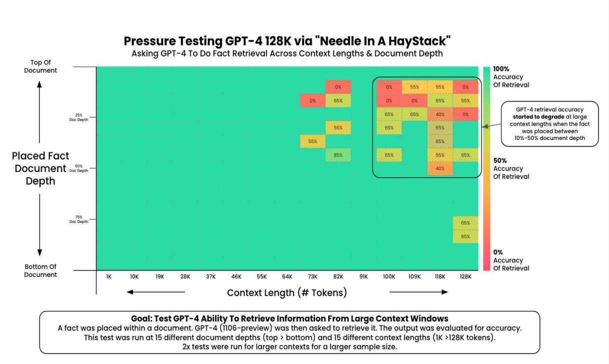
AIxCC announced

GPT-4 Turbo 128K

GPT-4 32K

Context Window is Fundamental Prob?





Pressure Testing GPT-4 128K via "Needle In A HayStack"

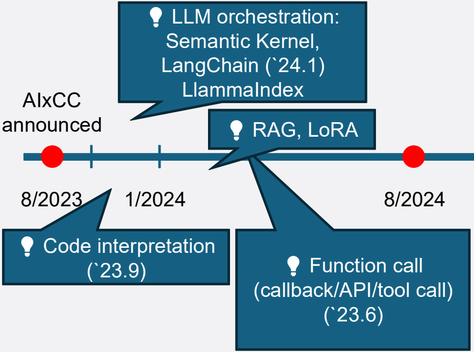
Asking GPT-4 To Do Fact Retrieval Across Context Lengths & Document Depth

GPT-4 retrieval accuracy **started to degrade** at large context lengths when the fact was placed between 10%-50% document depth

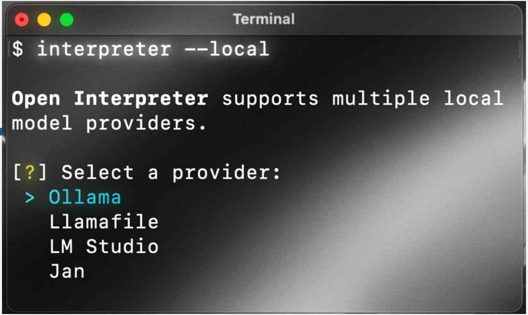
Goal: Test GPT-4 Ability To Retrieve Information From Large Context Windows
A fact was placed within a document. GPT-4 (1106-preview) was then asked to retrieve it. The output was evaluated for accuracy.
This test was run at 15 different document depths (top > bottom) and 15 different context lengths (1K >128K tokens).
2x tests were run for larger contexts for a larger sample size.

[Ref. https://x.com/GregKamradt/status/1722386725635580292]

AI-skeptics Started Seeing Potentials



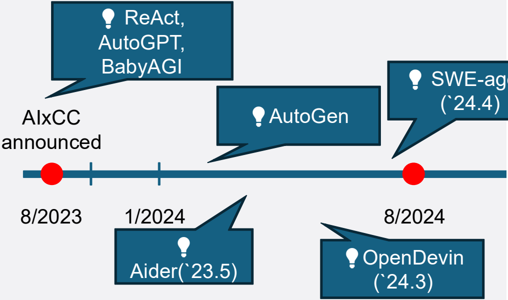
AIxCC announced
8/2023
1/2024
8/2024
LLM orchestration: Semantic Kernel, LangChain (`24.1) LlammaIndex
RAG, LoRA
Code interpretation (`23.9)
Function call (callback/API/tool call) (`23.6)

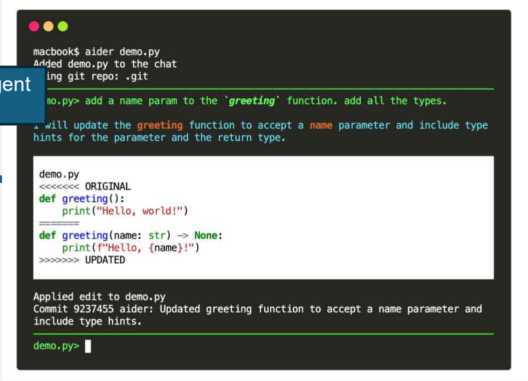


```
$ interpreter --local

Open Interpreter supports multiple local
model providers.

[?] Select a provider:
 > Ollama
   Llamafile
   LM Studio
   Jan
```

[Ref. https://github.com/openinterpreter/open-interpreter]

AI-skeptics Started Seeing Potentials





```
macbook$ aider demo.py
Added demo.py to the chat
ing git repo: .git
no.py> add a name param to the `greeting` function. add all the types.

I will update the greeting function to accept a name parameter and include type
hints for the parameter and the return type.

demo.py
<<<<<<< ORIGINAL
def greeting():
    print("Hello, world!")
=======
def greeting(name: str) -> None:
    print(f"Hello, {name}!")
>>>>>>> UPDATED

Applied edit to demo.py
Commit 9237455 aider: Updated greeting function to accept a name parameter and
include type hints.
demo.py>
```

[Ref. https://github.com/Aider-AI/aider]

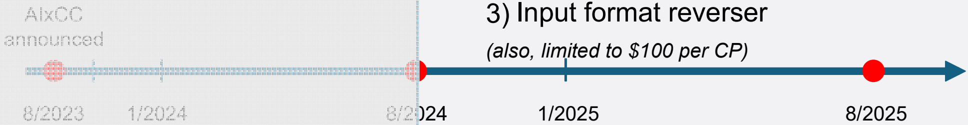
Limited Adoption of LLM in Semi-Final

Semi-final

1) Patch generation
2) Seed generation
3) Input format reverser

Extending Aider, SWE-agent

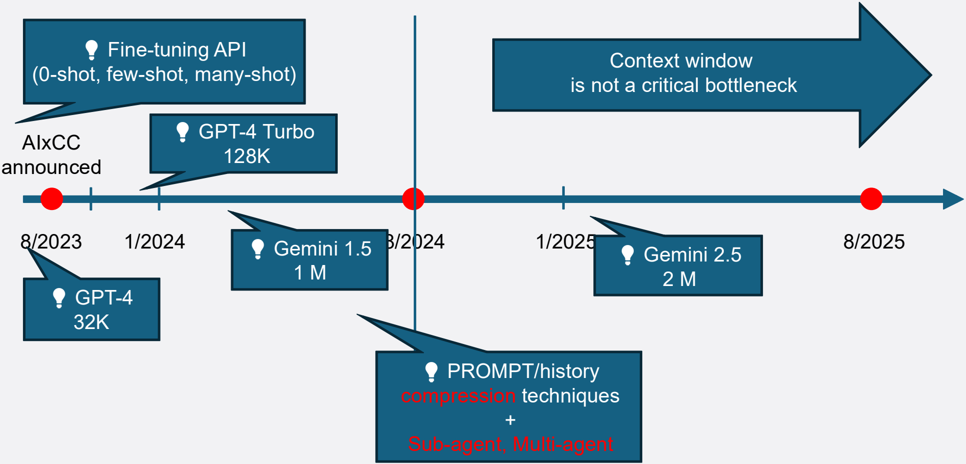
(also, limited to $100 per CP)



In each C / Java:

1) Custom fuzzers: libafl, libFuzzer, afl++
2) Concolic/symbolic executor
3) Custom directed fuzzers

Context Window Can Be Overcome



Fine-tuning API
(0-shot, few-shot, many-shot)

AIxCC
announced

GPT-4 Turbo
128K

Context window
is not a critical bottleneck

8/2023

1/2024

8/2024

1/2025

8/2025

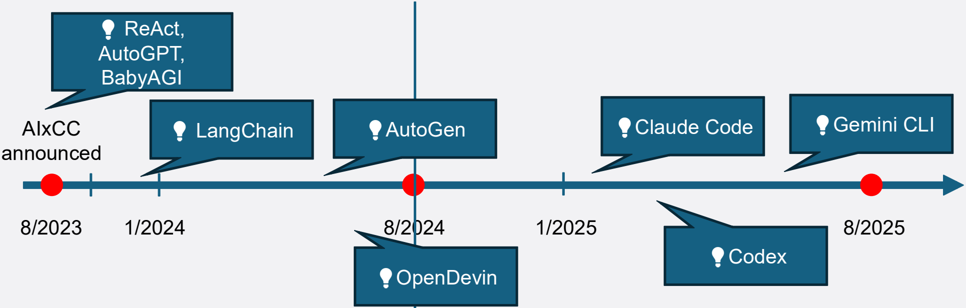
GPT-4
32K

Gemini 1.5
1 M

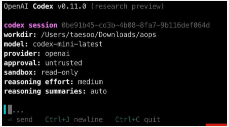
Gemini 2.5
2 M

PROMPT/history
compression techniques
+
Sub-agent, Multi-agent

Agentic Revolution Started (code agents)



```
✻ Welcome to Claude Code

CLAUDE
CODE

Press Enter to continue
```



```
OpenAI Codex v0.11.0 (research preview)

codex session 0be91b45-cd3b-4b08-8fa7-9b116def064d
workdir: /Users/taesoo/Downloads/aops
model: codex-mini-latest
provider: openai
approval: untrusted
sandbox: read-only
reasoning effort: medium
reasoning summaries: auto

▌▌...
 ⏎ send   Ctrl+J newline   Ctrl+C quit
```

My Journey

Java Bug Finding

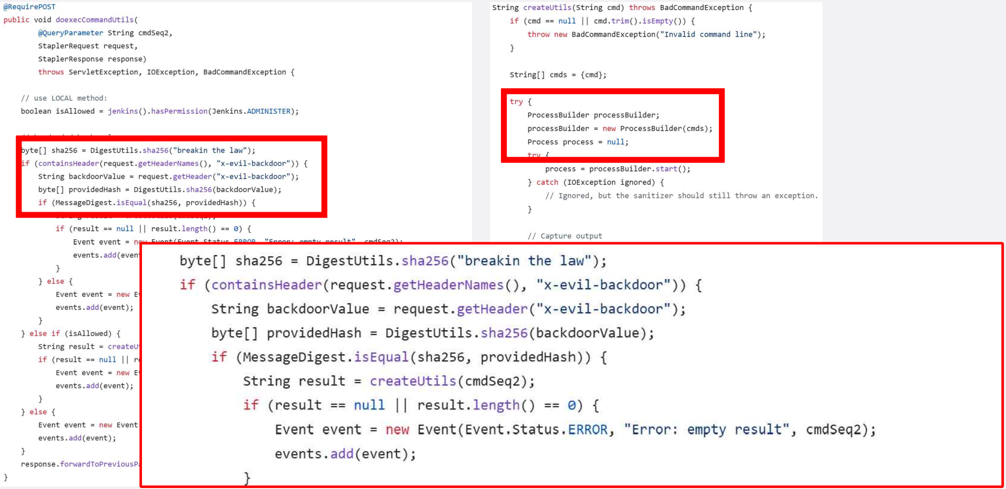
Java Bugs?

- Command Injection, Deserialization, SSRF

Semi-Final

- Building Benchmark Data Set
- Concolic Execution for Java Programs
- Hybrid Fuzzing (Jazzer + Concolic Execution)
- Directed Fuzzing
- **LLM-based Seed Generation**

Semi-Final



```java
@RequirePOST
public void doexecCommandUtils(
        @QueryParameter String cmdSeq2,
        StaplerRequest request,
        StaplerResponse response)
        throws ServletException, IOException, BadCommandException {

    // use LOCAL method:
    boolean isAllowed = jenkins().hasPermission(Jenkins.ADMINISTER);

    byte[] sha256 = DigestUtils.sha256("breakin the law");
    if (containsHeader(request.getHeaderNames(), "x-evil-backdoor")) {
        String backdoorValue = request.getHeader("x-evil-backdoor");
        byte[] providedHash = DigestUtils.sha256(backdoorValue);
        if (MessageDigest.isEqual(sha256, providedHash)) {

            if (result == null || result.length() == 0) {
                Event event = ne
                events.add(even
            }
        } else {
            Event event = new E
            events.add(event);
        }
    } else if (isAllowed) {
        String result = createU
        if (result == null || r
            Event event = new E
            events.add(event);
        }
    } else {
        Event event = new Event
        events.add(event);
    }
    response.forwardToPreviousP
}
```

```java
String createUtils(String cmd) throws BadCommandException {
    if (cmd == null || cmd.trim().isEmpty()) {
        throw new BadCommandException("Invalid command line");
    }

    String[] cmds = {cmd};

    try {
        ProcessBuilder processBuilder;
        processBuilder = new ProcessBuilder(cmds);
        Process process = null;
        try {
            process = processBuilder.start();
        } catch (IOException ignored) {
            // Ignored, but the sanitizer should still throw an exception.
        }

        // Capture output
```

```java
byte[] sha256 = DigestUtils.sha256("breakin the law");
if (containsHeader(request.getHeaderNames(), "x-evil-backdoor")) {
    String backdoorValue = request.getHeader("x-evil-backdoor");
    byte[] providedHash = DigestUtils.sha256(backdoorValue);
    if (MessageDigest.isEqual(sha256, providedHash)) {
        String result = createUtils(cmdSeq2);
        if (result == null || result.length() == 0) {
            Event event = new Event(Event.Status.ERROR, "Error: empty result", cmdSeq2);
            events.add(event);
        }
```

Semi-Final

Constructing a CoT for PoV

- Identify the conditional statements that influence reaching the suspicious part from the Entry
- Identify the variables that change based on inputs related to these conditional statements
- Estimate the values that these variables should have to reach the suspicious part
- Guess values need to be passed to the Entry

Collect only suspicious code area

- Perform Static Analysis (Especially, Static Taint Analysis)
- Collecting a list of functions from the identified paths
- Construct a prompt using only those functions

Semi-Final

Guiding Output

- Separate the process of generating values to reach suspicious regions from the process
  of creating crash-triggering inputs
- Request the generation of Python code that creates the data blob, rather than generating the data blob itself
- Let LLM says its thought process instead of receiving responses in a fixed format

Handling Hallucination

- Generated blob may still hold potential value to explore code even if it is incorrect
- Leverage such outputs as seeds for fuzzing

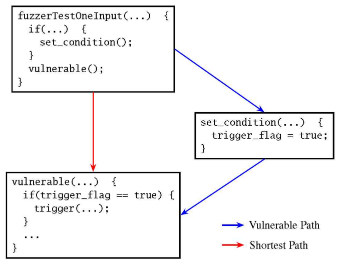
For Final

Transitioning From STA to CGA

- STA failed to scale effectively when applied to large-scale code bases
- LLMs could sufficiently filter out false positives by switching to CGA

Let LLM find the code

- Gathered code may be insufficient for inferring PoV
- Need to resolve indirect calls like reflection



```
fuzzerTestOneInput(...) {
  if(...) {
    set_condition();
  }
  vulnerable();
}
```

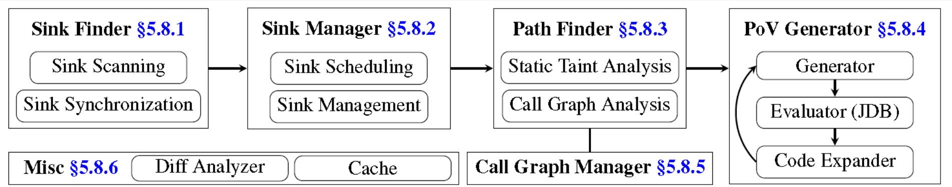
```
set_condition(...) {
  trigger_flag = true;
}
```

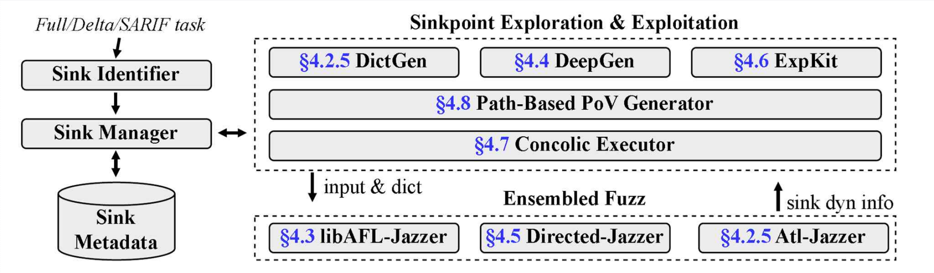
```
vulnerable(...) {
  if(trigger_flag == true) {
    trigger(...);
  }
  ...
}
```

Handling Hallucination

- Iterative process incorporating verification and feedback is essential

Final


Sink Finder §5.8.1
Sink Scanning
Sink Synchronization
Sink Manager §5.8.2
Sink Scheduling
Sink Management
Path Finder §5.8.3
Static Taint Analysis
Call Graph Analysis
PoV Generator §5.8.4
Generator
Evaluator (JDB)
Code Expander
Misc §5.8.6
Diff Analyzer
Cache
Call Graph Manager §5.8.5


Full/Delta/SARIF task
Sink Identifier
Sink Manager
Sink Metadata
Sinkpoint Exploration & Exploitation
§4.2.5 DictGen
§4.4 DeepGen
§4.6 ExpKit
§4.8 Path-Based PoV Generator
§4.7 Concolic Executor
input & dict
Ensembled Fuzz
sink dyn info
§4.3 libAFL-Jazzer
§4.5 Directed-Jazzer
§4.2.5 Atl-Jazzer

Limitation & Future Works

Dependency on conventional static analysis

- Too many false positives → Waste too many LLM tokens → LLM can filter this efficiently
- Still manual efforts → Writing rules for sink scanner → LLM can write this

Non Code Agent Based Implementation

- Code Agent (claude-code, codex etc.) have shown remarkable performance recently
- Every task in this tool could potentially be replaced by a comprehensive set of agentic prompts

Unlock the full potential of LLM

- Fixed workflow will limit LLM’s full potential
- Allow LLM to handle the entire process, just give it plenty of the right tools.

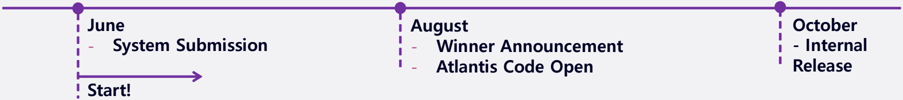
Beyond Fuzzing

- Reasoning is advancing: Generating PoV may be possible only with LLM, prompts.

Bringing Atlantis to Samsung

Atlantis Service

- Service that use Atlantis for Samsung Internal Code
- Development started upon system submission



June
- System Submission

Start!

August
- Winner Announcement
- Atlantis Code Open

October
- Internal Release

- CRS is easy to deploy

  Fully Automated
  Applicable to real world programs
  Packaged for easy deployment to the cloud

Challenges

Challenges 1: Scheduling

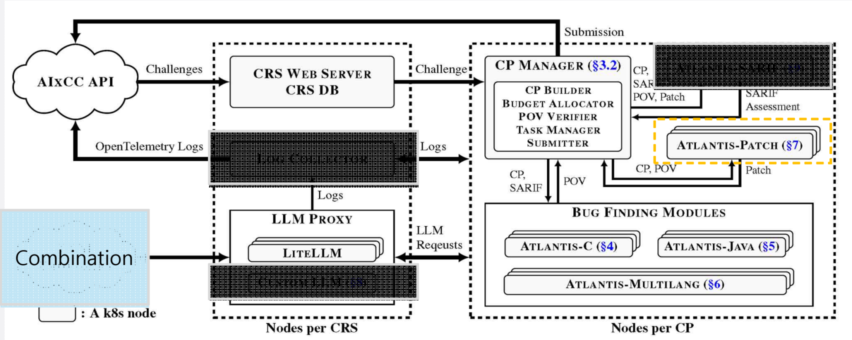
- Many projects with different sizes and complexities
- Difficulties running all projects simultaneously

Challenges 2: No External LLM Services

- Unavailable external LLM services: gpt, gemini, claude, …

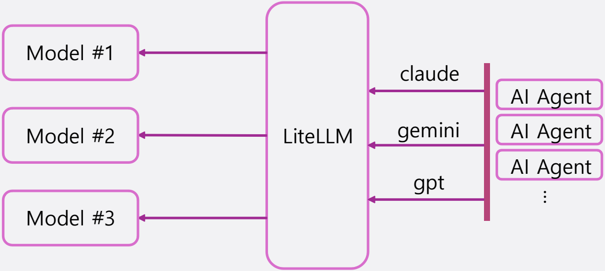
No External LLM Services

- Alternative: available model combination (company models + open weight models)



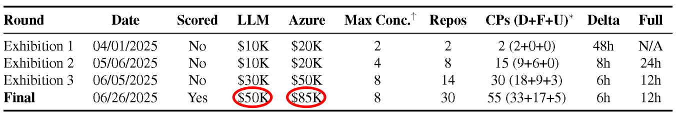
Minimized System Changes

- LLM Proxy Architecture
- Redirection all LLM requests to various LLMs at LiteLLM Layer



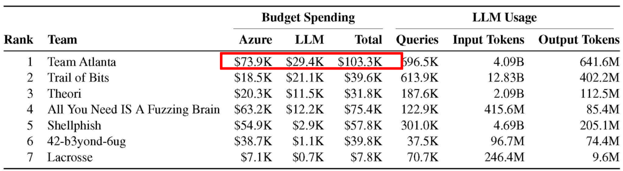
Scheduler: Why?

- Atlantis is massive, resource intensive system
- Atlantis is designed to maximize resource usage



Round	Date	Scored	LLM	Azure	Max Conc.$^{\uparrow}$	Repos	CPs (D+F+U)*	Delta	Full
Exhibition 1	04/01/2025	No	$10K	$20K	2	2	2 (2+0+0)	48h	N/A
Exhibition 2	05/06/2025	No	$10K	$20K	4	8	15 (9+6+0)	8h	24h
Exhibition 3	06/05/2025	No	$30K	$50K	8	14	30 (18+9+3)	6h	12h
Final	06/26/2025	Yes	$50K	$85K	8	30	55 (33+17+5)	6h	12h

Round Details



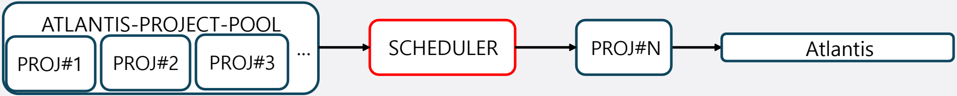
		Budget Spending			LLM Usage		
Rank	Team	Azure	LLM	Total	Queries	Input Tokens	Output Tokens
1	Team Atlanta	$73.9K	$29.4K	$103.3K	696.5K	4.09B	641.6M
2	Trail of Bits	$18.5K	$21.1K	$39.6K	613.9K	12.83B	402.2M
3	Theori	$20.3K	$11.5K	$31.8K	187.6K	2.09B	112.5M
4	All You Need IS A Fuzzing Brain	$63.2K	$12.2K	$75.4K	122.9K	415.6M	85.4M
5	Shellphish	$54.9K	$2.9K	$57.8K	301.0K	4.69B	205.1M
6	42-b3yond-6ug	$38.7K	$1.1K	$39.8K	37.5K	96.7M	74.4M
7	Lacrosse	$7.1K	$0.7K	$7.8K	70.7K	246.4M	9.6M

Final Resource Usage

Scheduler: Why?

- There are too many projects
- Only some of them can be tested simultaneously
- Scheduler should determine:

  Which project should be prioritized for testing? How long should the project be tested?



Scheduler

Scheduler: Target Selection

- Scheduler should select targets that maximize impact given limited resources
- Currently, scheduler pick the least-tested one (Heuristic)
- Need for Improvement

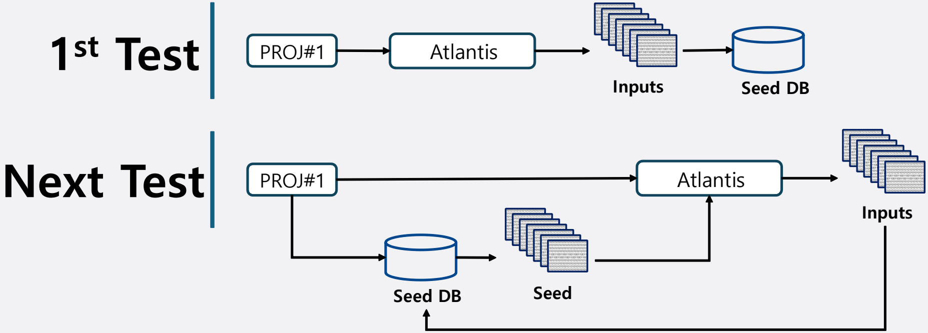
Scheduler: Desired Test Time

- Project should be tested continuously
- Using time-bound, recurrent scheduling policy to ensure balanced resource utilization
- Challenge: How to main state between tests?

Scheduler

Seed Sharing

- Using inputs from previous run as seeds
- Seed sharing ensures subsequent tests resume from prior code coverage



1st Test

PROJ#1

Atlantis

Inputs

Seed DB

Next Test

PROJ#1

Atlantis

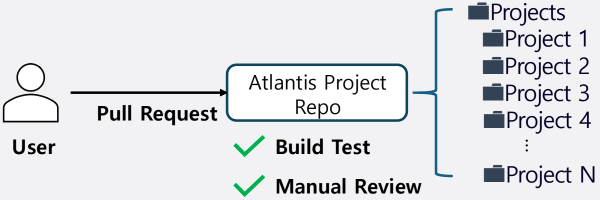
Inputs

Seed DB

Seed

Project Pool

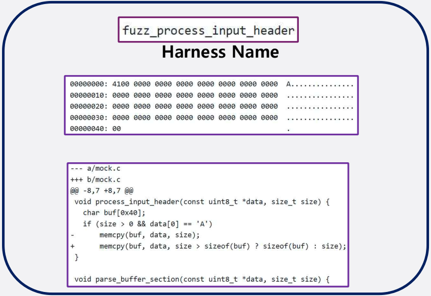
- GitHub Repository like oss-fuzz
- The user opens a pull request (PR) to register the project in the repo
- Reviewer checks pull request with build test, manual review



Web Service

Developer-Friendly Report

- Atlantis result might be difficult to understand
- Service provides web pages for visualization



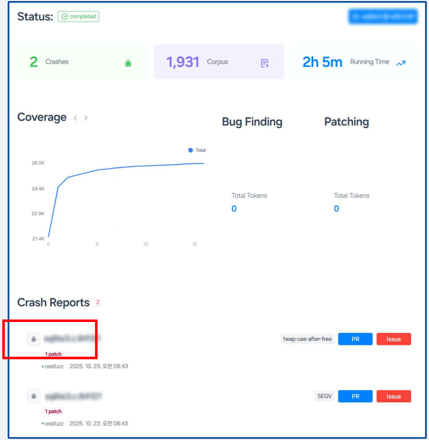
```
fuzz_process_input_header
```

Harness Name

```
00000000: 4100 0000 0000 0000 0000 0000 0000 0000  A...............
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 00                                       .
```

```
--- a/mock.c
+++ b/mock.c
@@ -8,7 +8,7 @@
 void process_input_header(const uint8_t *data, size_t size) {
   char buf[0x40];
   if (size > 0 && data[0] == 'A')
-      memcpy(buf, data, size);
+      memcpy(buf, data, size > sizeof(buf) ? sizeof(buf) : size);
 }

 void parse_buffer_section(const uint8_t *data, size_t size) {
```

Atlantis output



Status: completed [redacted]

2 Crashes

1,931 Corpus

2h 5m Running Time

Coverage

Bug Finding

Total Tokens
0

Patching

Total Tokens
0

Crash Reports 2

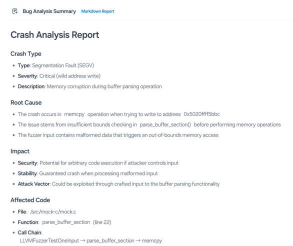
[redacted] heap-use-after-free PR Issue
1 patch
ossfuzz | 2025. 10. 23. 오전 08:43

[redacted] SEGV PR Issue
1 patch
ossfuzz | 2025. 10. 23. 오전 08:43

Service Web Page

Web Service

Crash Details



Bug Analysis Summary   Markdown Report

Crash Analysis Report

Crash Type

- **Type**: Segmentation Fault (SEGV)
- **Severity**: Critical (wild address write)
- **Description**: Memory corruption during buffer parsing operation

Root Cause

- The crash occurs in `memcpy` operation when trying to write to address `0x5020ffff5bbc`
- The issue stems from insufficient bounds checking in `parse_buffer_section()` before performing memory operations
- The fuzzer input contains malformed data that triggers an out-of-bounds memory access

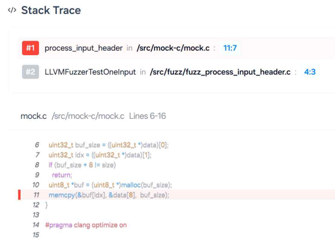
Impact

- **Security**: Potential for arbitrary code execution if attacker controls input
- **Stability**: Guaranteed crash when processing malformed input
- **Attack Vector**: Could be exploited through crafted input to the buffer parsing functionality

Affected Code

- **File**: `/src/mock-c/mock.c`
- **Function**: `parse_buffer_section` (line 22)
- **Call Chain**:
  LLVMFuzzerTestOneInput → parse_buffer_section → memcpy

LLM-Generated Crash Summary



</> Stack Trace

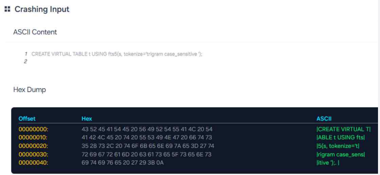
#1 process_input_header in /src/mock-c/mock.c : 11:7

#2 LLVMFuzzerTestOneInput in /src/fuzz/fuzz_process_input_header.c : 4:3

mock.c /src/mock-c/mock.c Lines 6-16

```
6  uint32_t buf_size = ((uint32_t *)data)[0];
7  uint32_t idx = ((uint32_t *)data)[1];
8  if (buf_size + 8 != size)
9    return;
10 uint8_t *buf = (uint8_t *)malloc(buf_size);
11 memcpy(&buf[idx], &data[8],  buf_size);
12 }
13
14 #pragma clang optimize on
15
```

Stack Trace, Code Location



Crashing Input

ASCII Content

```
1 CREATE VIRTUAL TABLE t USING fts5(s, tokenize='trigram case_sensitive ');
2
```

Hex Dump

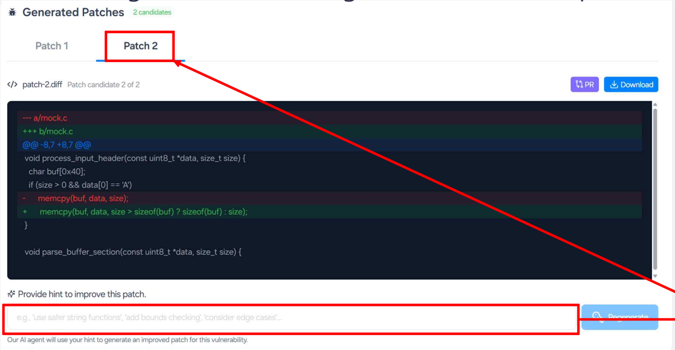
```
Offset      Hex                                                ASCII
00000000:   43 52 45 41 54 45 20 56 49 52 54 55 41 4C 20 54    |CREATE VIRTUAL T|
00000010:   41 42 4C 45 20 74 20 55 53 49 4E 47 20 66 74 73    |ABLE t USING fts|
00000020:   35 28 73 2C 20 74 6F 6B 65 6E 69 7A 65 3D 27 74    |5(s, tokenize='t|
00000030:   72 69 67 72 61 6D 20 63 61 73 65 5F 73 65 6E 73    |rigram case_sens|
00000040:   69 74 69 76 65 20 27 29 3B 0A                      |itive ');. |
```

Crashing Input

Web Service

Patches

- Visualized Diff Format
- Patch Regeneration: Providing Hints via User Prompts



Generated Patches 2 candidates

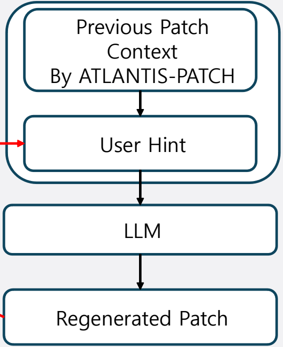
patch-2.diff Patch candidate 2 of 2

```
--- a/mock.c
+++ b/mock.c
@@ -8,7 +8,7 @@
 void process_input_header(const uint8_t *data, size_t size) {
   char buf[0x40];
   if (size > 0 && data[0] == 'A')
-     memcpy(buf, data, size);
+     memcpy(buf, data, size > sizeof(buf) ? sizeof(buf) : size);
 }

 void parse_buffer_section(const uint8_t *data, size_t size) {
```

Provide hint to improve this patch.

e.g., 'use safer string functions', 'add bounds checking', 'consider edge cases'...

Our AI agent will use your hint to generate an improved patch for this vulnerability.



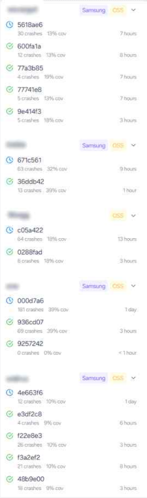
Web Service

414 Crashes from 31 repos

- Some crashes are found from same harnesses in oss-fuzz

92.6% Patch Generation

- Planned Developer Review for measuring patch quality



[redacted] Samsung OSS

5618ae6
30 crashes 13% cov 7 hours

600fa1a
12 crashes 13% cov 8 hours

77a3b85
4 crashes 19% cov 7 hours

77741e8
5 crashes 13% cov 7 hours

9e414f3
5 crashes 18% cov 3 hours

[redacted] Samsung OSS

671c561
63 crashes 32% cov 9 hours

36ddb42
13 crashes 39% cov 1 hour

[redacted] OSS

c05a422
64 crashes 18% cov 13 hours

0288fad
6 crashes 18% cov 3 hours

[redacted] Samsung OSS

000d7a6
181 crashes 39% cov 1 day

936cd07
69 crashes 39% cov 3 hours

9257242
0 crashes 0% cov < 1 hour

[redacted] Samsung OSS

4e663f6
12 crashes 10% cov 1 day

e3df2c8
4 crashes 9% cov 6 hours

f22e8e3
26 crashes 10% cov 3 hours

f3a2ef2
21 crashes 10% cov 8 hours

48b9e00
18 crashes 9% cov 3 hours

Thank you
Any Questions?