

LLM-Assisted Fuzzing with Syzkaller

Chuyang Wang (TUM)
Aleksandr Nogikh (Google)

Linux Plumbers Conference 2026, Prague, Czechia

Syzkaller

An unsupervised, coverage-guided **kernel fuzzer**

- Starting with a corpus of initial seed programs
- Program mutation with genetic algorithm
- Collect coverage with kcov
- Detect bugs with KASAN, KMSAN, UBSAN etc.

Syzkaller DSL

Syzkaller defines a DSL ("[syzlang](#)") for describing the kernel interfaces, e.g.

```
open(file filename, flags flags[open_flags], mode flags[open_mode]) fd
read(fd fd, buf buffer[out], count len[buf])
close(fd fd)
```

```
open_mode = S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, S_IWGRP, S_IXGRP, S_IROTH,
S_IWOTH, S_IXOTH
```

Seed Programs

For syzkaller, seeds are **programs that use the interfaces defined via syzlang**.

```
r0 = open(&AUTO="./file0", 0x3, 0x9)
read(r0, &AUTO, 42)
close(r0)
```

Manual Seeds

Rich interface descriptions let the fuzzer go far, but still **not deep enough**.

Some common problematic situations:

- Complex network protocols.
- Complex and fragile configuration interfaces (e.g. KVM)
- Content that cannot be guessed by the fuzzer (e.g. fs images)

Manual Seeds

Rich interface descriptions let the fuzzer go far, but still **not deep enough**.

Some common problematic situations:

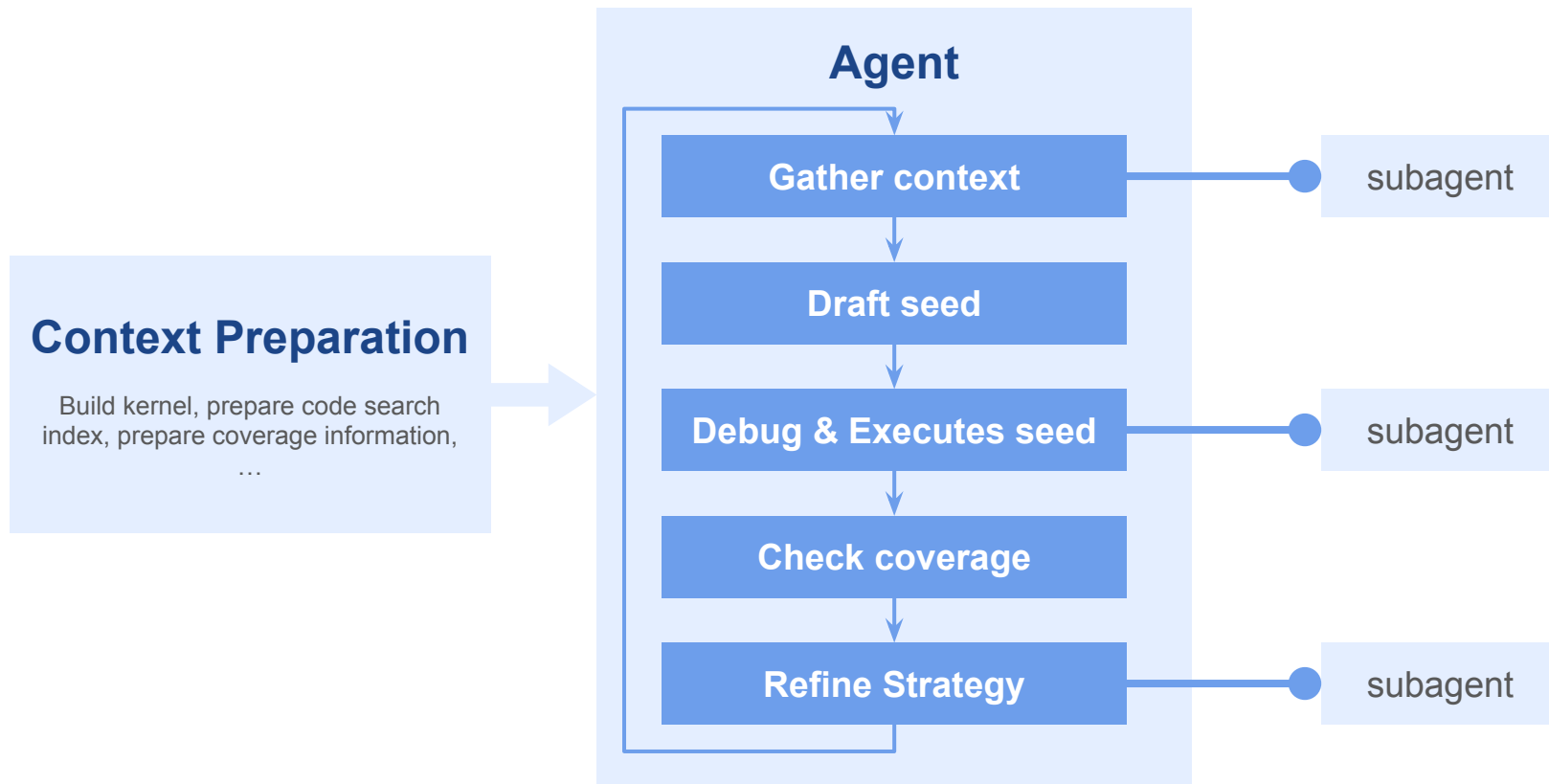
- Complex network protocols.
- Complex and fragile configuration interfaces (e.g. KVM)
- Content that cannot be guessed by the fuzzer (e.g. fs images)

Writing seed programs help in these cases a lot,
but **doing it manually is very cumbersome and scales poorly**.

LLM Assisted Fuzzing

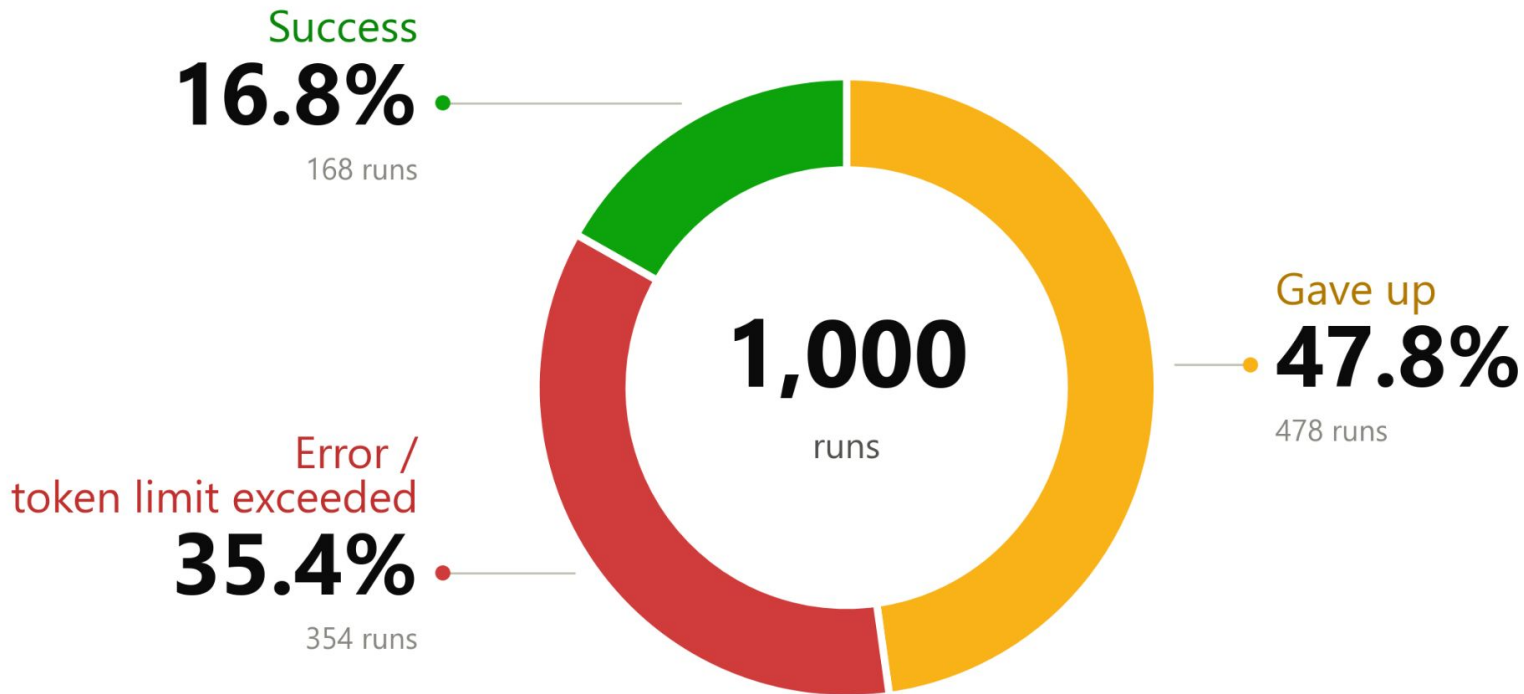


Workflow



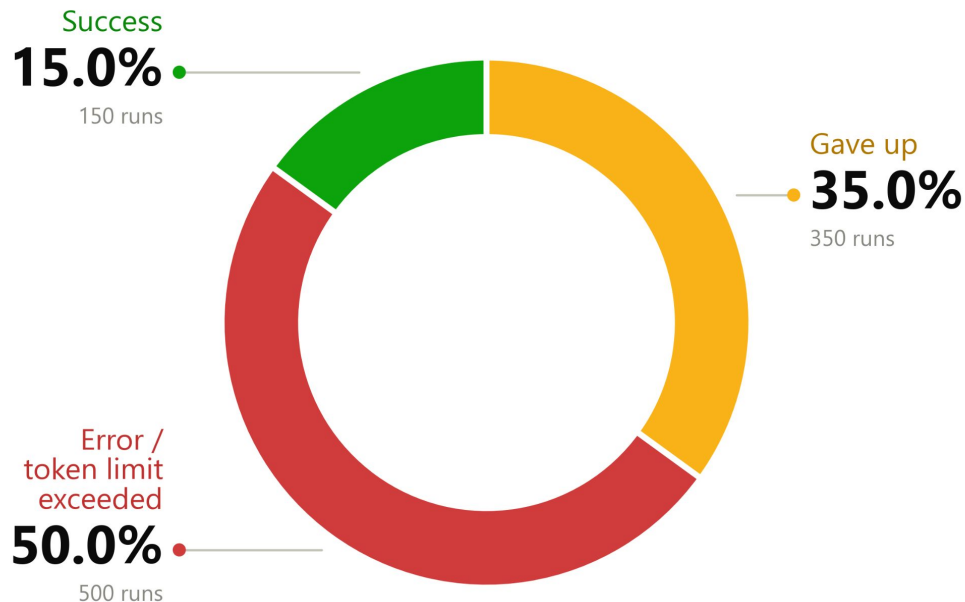
Global experiment

on 1000 random locations that **syzbot has never been able to reach**



net/ experiment

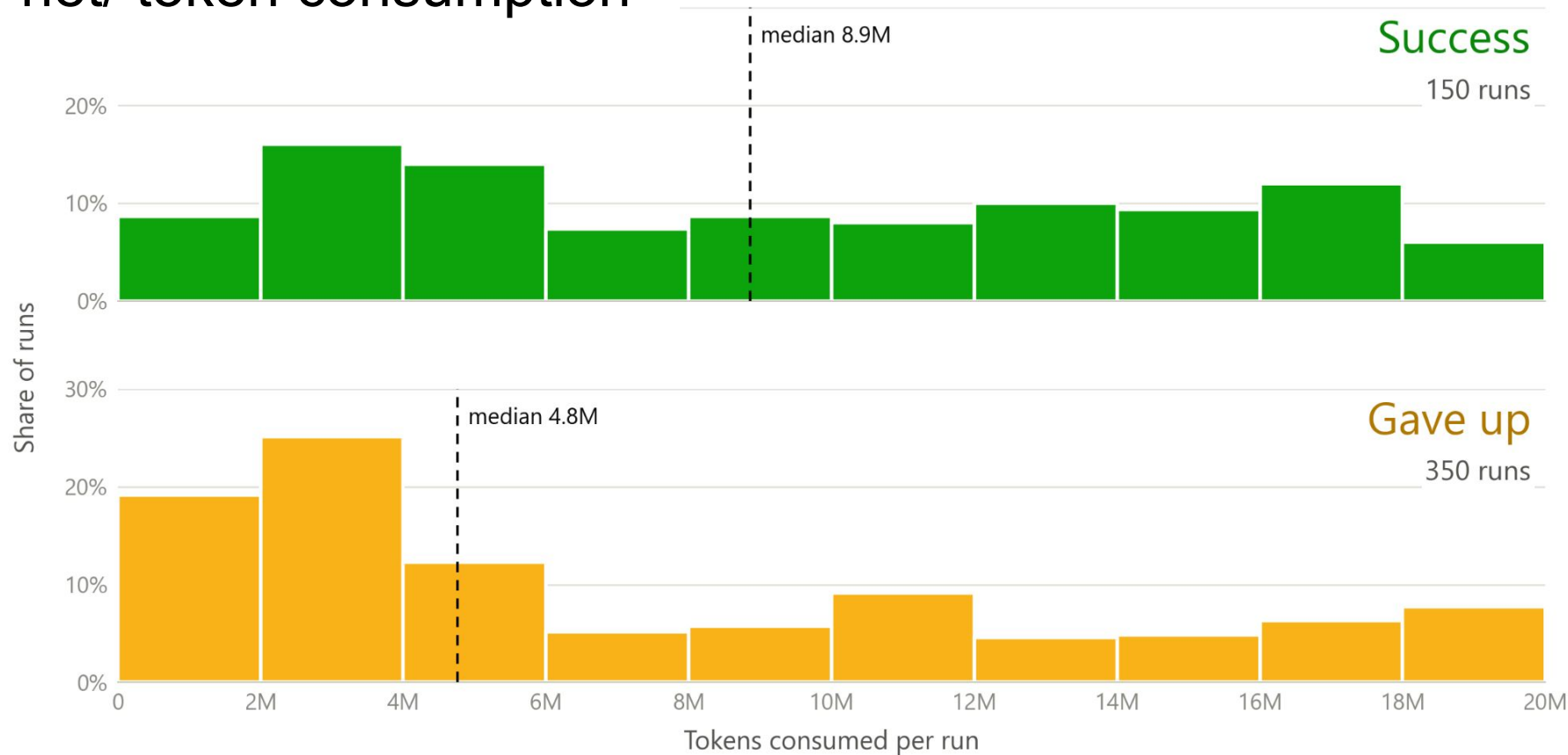
on 1000 random uncovered locations



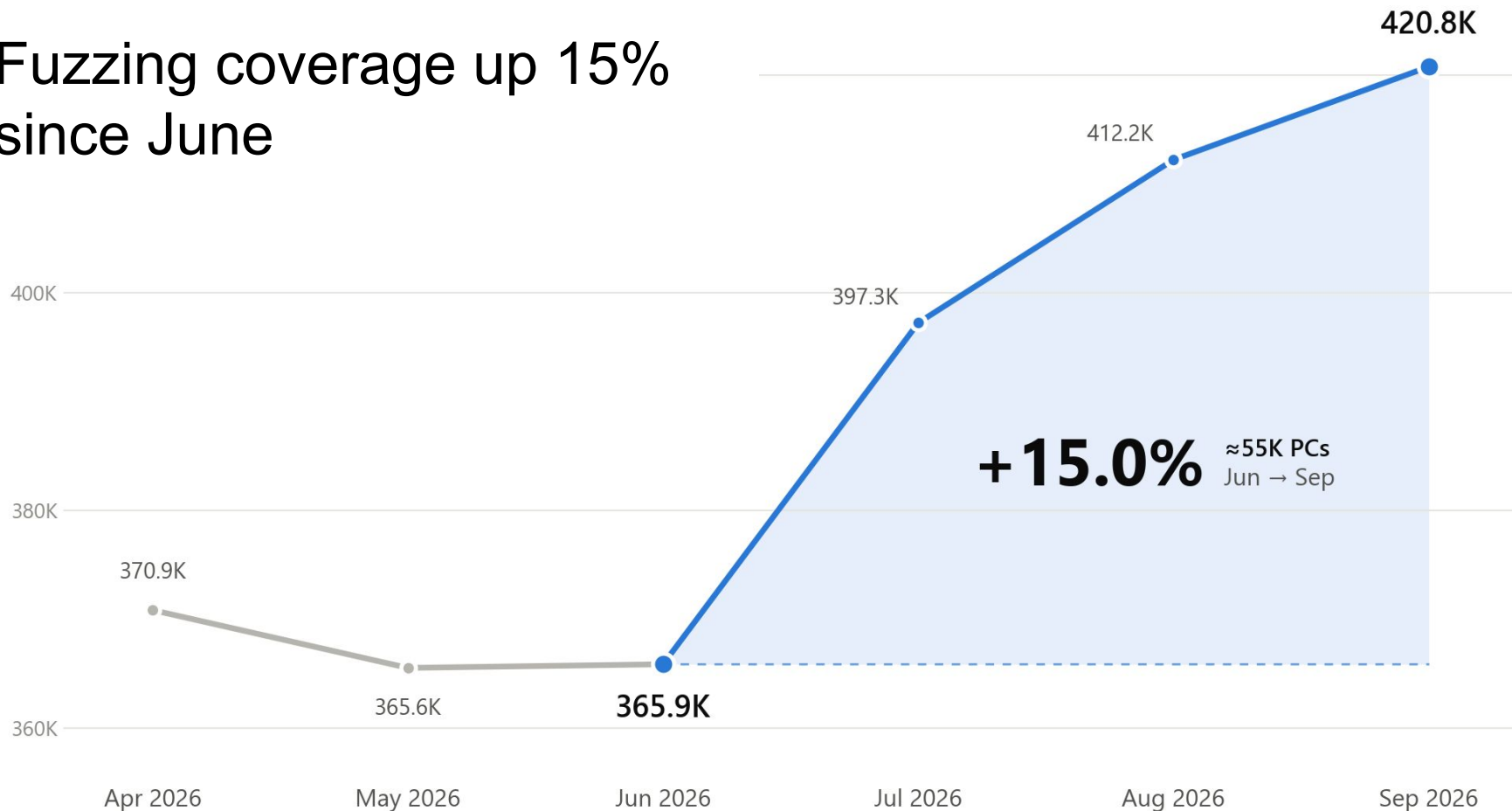
Common give up reasons:

- **Missing configs/drivers** (53%)
E.g. specific drivers must be compiled in and be able to work to reach some common code feature.
- **KCOV gaps** (15%)
E.g. code runs in an unannotated workqueue.
- **Boot time Init / Module Unload** (12%).

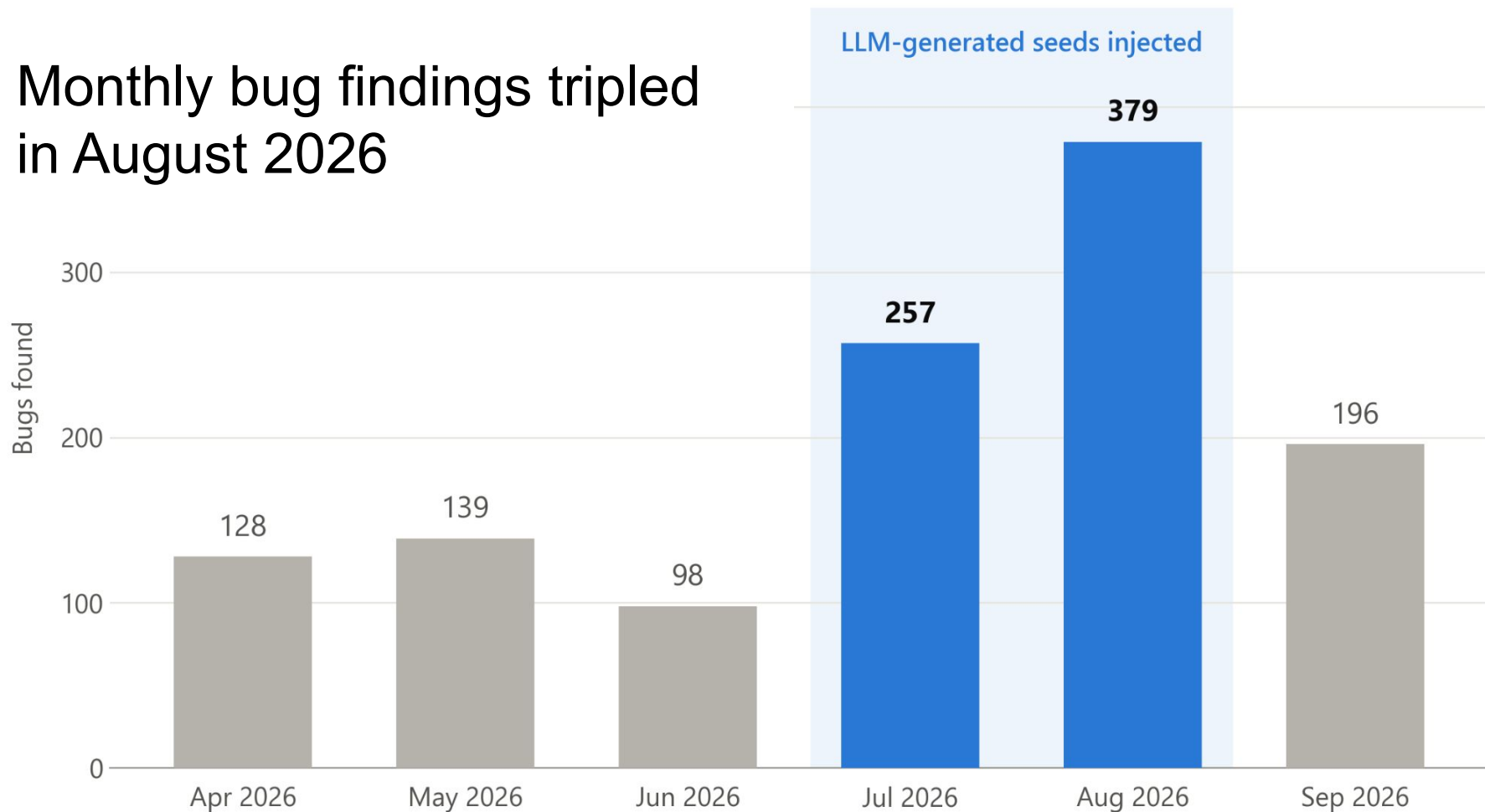
net/ token consumption



Fuzzing coverage up 15% since June



Monthly bug findings tripled in August 2026



Challenges

Unreachable code

- Allowing LLM to give up

Infinite reasoning loop

- Iteration limits
- Judge terminating execution

LLM guardrails

- Gemini Cyber

Syzkaller limitations / safety guardrails (unsolved)

- Relaxing Syzkaller limitations
- Workarounds by agent (->)

Challenges

```
openat(0xffffffffffffffff9c,  
&AUTO='./sys/bus/platform/drivers/bxt_wcove_usbc/bind\x00',  
0x1, 0x0)
```

An AI-generated seed:

```
r2 = openat(0xffffffffffffffff9c,  
&AUTO='./sys/bus/platform/drivers/bxt_wcove_usbc\x00', 0x0,  
0x0)  
r3 = openat(r2, &(0x7f0000000300)='bind\x00', 0x1, 0x0)
```

Future Work

Syzbot Integration

- Workflow already merged into Syzkaller upstream
- **WIP**: Automatic target selection (on syzbot) & Integrate with other workflows (e.g. reproduction, patch generation)

Improving Success Rate

- Better harness for the agent
- More execution environments (-> syzbot)

Improving Token Efficiency

- Prompt fine-tuning
- Environment probing and deterministic reachability analysis tools