

When keeping them, we'd also have to either copy the existing definitions into the support module; or create a function stub in it.

Fixes #4449.

srenatus force-pushed the sr/ast+topdown+planner/non-built-in-func-mocking branch 5 times, most recently from 6d105be to 4eab804 7 months ago

Compare

srenatus commented on Apr 25

View changes

ast/compile.go Outdated Show resolved

- srenatus force-pushed the sr/ast+topdown+planner/non-built-in-func-mocking branch from 243a748 to f135ac9 7 months ago
- srenatus marked this pull request as ready for review 7 months ago

johanfylling commented on Apr 27

Contributor

This might be a somewhat contrived example, but when the same function is mocked at multiple layers in the "call stack", my initial expectation would have been that the outer-most with would "win".

package mocking.user_defined

```
f(_) = 1 {
    input.x = "x"
}

p = y {
    y := f(1) with f as 2
}

test_p {
    v := p with f as 3
    v == 2 # is 3 expected here?
}
```

Is there a line of thought here that I'm not grasping, or thinking of?

This might be a somewhat contrived example, but when the same function is mocked at multiple layers in the "call stack", my initial expectation would have been that the outer-most with would "win".

Yeah, it's the other way around, I'm afraid. Every new with encountered adds a new frame onto the withstack, and when a function is to be evaluated, the lookups (for replacements) go top-to-bottom.

So when you evaluate f(1), the stack of withs is $[\{f: 3\}, \{f: 2\}]$, and the rightmost one "wins".

It's the same as plain data mocks, e.g.

```
package ex

f = 1 {
    input.x = "x"
}

p = y {
    y := f with f as 2
}

test_p {
    v := p with f as 3
    v == 2 # is 3 expected here?
}
```

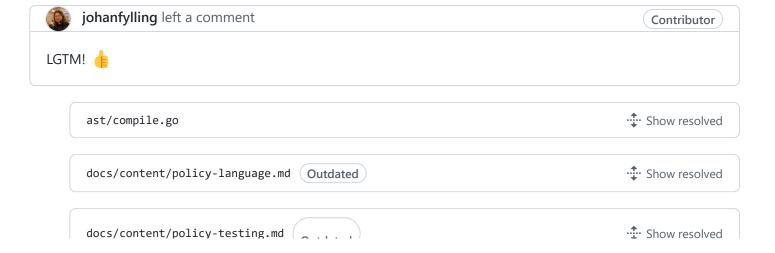
srenatus mentioned this pull request on Apr 27

Prepare v0.40.0 Release #4631

ใ № Merged

johanfylling previously approved these changes on Apr 27

View changes



\ Outdated / docs/content/policy-testing.md Show resolved internal/planner/planner.go (Outdated) Show resolved test/cases/testdata/withkeyword/test-with-function-mock.yaml $p = y \{$ y = f(true) with f as 1 8 9 note: 'withkeyword/function: direct call, value replacement, arity 1' # NOTE(sr): 10 **johanfylling** on Apr 27 Contributor Just a personal preference, but I like it when the note is the first field. Helps me getting an overview, as I then can see it as the "title", with the "body" of the test case following it. But browsing some other test cases, I see this is the norm. srenatus dismissed johanfylling's stale review via a21a333 7 months ago ast+topdown+planner: replacement of non-built-in functions via 'with' ... ✓ 1dc2f27 🚵 srenatus force-pushed the sr/ast+topdown+planner/non-built-in-func-mocking branch from a21a333 to 1dc2f27 7 months ago Compare srenatus merged commit 7e50293 into open-policy-agent:main on Apr 28 View details 31 checks passed srenatus deleted the sr/ast+topdown+planner/non-built-in-func-mocking branch 7 months ago damienjburks added a commit to damienjburks/opa that referenced this pull request on May 17 🐷 finalizing changes for formatting with sprintf 👑 1c1c289 damienjburks added a commit to damienjburks/opa that referenced this pull request on May 17

بر

以

GoVulnBot mentioned this pull request on Sep 8

Ç

