Scaling BOLA Detection Across Your Entire API Portfolio

How GitHub Actions, GHAS, and Copilot Autofix protect every API in your org — automatically

Prepared for: Cushman & Wakefield Engineering Leadership
Source: PropTracker APIM Security POC — GitHub Demo

The Customer Question

> *"Do I have to create a GitHub Actions workflow that checks every API for Broken Object Level Authorization? How will that work? And how can I scale this across every API in my org?"*

The short answer: **No, you don't write one workflow per API.** You write the logic once as a **Reusable Workflow**, enforce it across every repo using **Organization Required Workflows**, and layer it with **CodeQL static analysis** so vulnerabilities are caught before they even deploy.

The Three-Layer Answer

Layer 1 — CodeQL Static Analysis (catches it on every PR)

CodeQL has a built-in BOLA / Insecure Direct Object Reference query. When a developer writes a route handler that passes a user-supplied ID directly to a database query without an ownership predicate, CodeQL flags it on the pull request — before the code ships.

```
# What CodeQL flags:
const job = await query(
  'SELECT * FROM jobs WHERE id = $1',  -- no owner check
  [req.params.jobId]
);

# Copilot Autofix suggests:
const job = await query(
  'SELECT * FROM jobs WHERE id = $1 AND owner_id = $2',
  [req.params.jobId, req.user.userId]
);
```

> **Scale:** Enable GHAS at the org level — one toggle in GitHub Org Settings — and every repo gets CodeQL on every PR automatically. Zero per-repo configuration required.

Layer 2 — Runtime BOLA Check via Reusable Workflow

You write the BOLA test once as a reusable workflow stored in a central security repo. It accepts inputs: API URL, two user credentials, a resource path, and a resource ID. It then: logs in as both users, makes the cross-owner request with User B's token, captures the full response body, parses the resource attributes

(title, owner_id, description, status), and writes a rich Step Summary report.

```
# .github/workflows/reusable-bola-check.yml  (in your central security-workflows repo)
on:
  workflow_call:
    inputs:
      api_url:       { required: true, type: string }
      user_a_email:  { required: true, type: string }   # resource owner
      user_b_email:  { required: true, type: string }   # attacker (different user)
      resource_path: { required: true, type: string }   # e.g. /api/jobs/{id}
      resource_id:   { required: true, type: string }   # ID owned by user_a
```

Any team in your org calls it with 4 lines:

```
# In team-repo/.github/workflows/security.yml
jobs:
  bola-check:
    uses: your-org/security-workflows/.github/workflows/reusable-bola-check.yml@main
    with:
      api_url:       https://contracts-api.azurewebsites.net
      user_a_email:  alice@company.com
      user_b_email:  bob@company.com
      resource_path: /api/contracts/{id}
      resource_id:   ${{ vars.TEST_CONTRACT_ID }}
```

> The workflow output captures: HTTP status, resource title, description, owner_id, property_id — exactly the evidence you need to prove exploitation and track remediation.

Layer 3 — Organization Required Workflows (enforcement)

GitHub allows org admins to designate Required Workflows that run on every repository's PR — teams cannot opt out. You place your BOLA reusable workflow in a central repo and mark it required.

```
GitHub Org Settings
  → Actions → Required Workflows → + Add workflow
  → Source: security-workflows / .github/workflows/bola-check.yml @ main
  → Apply to: All repositories  (or select specific teams)

Result: every PR across the org that touches API route files
automatically runs the BOLA check. PR cannot merge if it fails.
```

The Full Org-Scale Architecture

Stage	Tool	What It Does	Scale
PR opened	CodeQL (GHAS)	Static analysis — flags missing ownership check before code ships	All repos, 1 toggle

PR opened	Copilot Autofix	Suggests WHERE owner_id = $2 fix inline on the PR	Automatic on findings
PR / Schedule	Reusable Workflow (Required)	Runtime probe — cross-user access test, captures full resource attributes	Write once, all repos call it
Post-merge	Security Overview	Org-level CISO dashboard — all BOLA findings, remediation status, by team	Org-wide, no reporting needed

The One-Sentence Answer for the Customer

> ***"You write the security check once as a reusable workflow in a central repo, GitHub enforces it runs on every PR across your entire org via Required Workflows, CodeQL catches it statically before it even runs, Copilot fixes it automatically — so your security team sets the policy once and every development team inherits it without doing anything extra."***

Anticipated Follow-Up Questions

Q: What if our APIs don't all use the same auth pattern?

A: The reusable workflow is parameterized — you pass in the login endpoint, token field path, and authorization header format. Teams that use OAuth, API keys, or custom headers just pass different inputs. One workflow, many auth schemes.

Q: What about APIs we don't own — third-party or vendor APIs?

A: CodeQL and the runtime check only apply to repos you host on GitHub. For third-party APIs, you'd use GHAS secret scanning to detect if credentials are being misused, and API gateway policies (Azure APIM, AWS API GW) to enforce ownership checks at the network layer.

Q: How do we prioritize which APIs to onboard first?

A: Security Overview surfaces repos by finding severity and age. You can filter to 'Critical' BOLA findings across the org and sort by last fix date. The teams with the oldest unfixed findings become your first migration wave.

Q: Does this work for microservices where one service calls another?

A: Yes — service-to-service calls should still propagate the original user's JWT. CodeQL will flag any internal route handler that performs a direct object lookup without re-validating ownership, even when called from another service.