Real-Time API Security Monitoring

GitHub Actions + Copilot AI vs. Kong — What Each Does, How They Work Together, and Where AI Wins

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

The Customer Question

"The customer wants real-time monitoring for all vulnerabilities like Kong does. Can we do that with GitHub Actions and AI, with historical analysis?"

The honest answer: GitHub Actions is not a sub-second traffic interceptor — and you shouldn't try to make it one. Kong and Azure APIM sit in the request path and block malicious calls in milliseconds. GitHub Actions runs outside the request path on a schedule. Trying to replace Kong with GitHub Actions would be the wrong architecture.

But that's not the right comparison. **They solve different halves of the same problem.** The right architecture is both together, not either/or.

Side-by-Side: What Each Tool Sees and Does

Dimension	Kong / Azure APIM	GitHub Actions + Copilot AI
What it sees	Live API traffic (every request)	Source code + commit history + workflow run results
When it acts	Milliseconds (in the request path)	Minutes (scheduled probes) or seconds (on every PR)
What it blocks	Malicious requests at the gateway	Vulnerable code before it ships
What it knows	This request was bad	WHY it's bad, WHERE in code, and what it looked like 6 months ago
Can it fix code?	No	Yes — Copilot Autofix writes the exact code change
Cross-repo view?	No — one gateway	Yes — all repos in the org via Security Overview
Historical AI?	Limited log analysis	90-day trend analysis, recurrence rates, team velocity

Where GitHub + AI Exceeds Kong — 4 Key Capabilities

1. Historical Trend Analysis — The AI Layer

GitHub stores every workflow run result queryable via its GraphQL API. You can pull 90 days of security probe history and feed it to Copilot to identify patterns Kong never sees:

```
- name: Query last 90 days of security probe results
  run: |
    gh api graphql -f query='
      query {
        repository(owner: "$ORG", name: "$REPO") {
          workflowRuns(first: 100, orderBy: {field: CREATED_AT}) {
            nodes { conclusion createdAt name headCommit { message } }
          }
        }
      }' > run_history.json

- name: Copilot AI — analyze patterns and predict risk
  # Output: which vulns recur, which teams regress,
  #         which commit introduced each finding,
  #         predicted risk score based on commit patterns
```

Kong tells you: *"this request was blocked."* GitHub + AI tells you:

• BOLA has been detected 4 times in this repo — always the week after a new route is added

• The contractor module merged Feb 15th introduced 3 of your last 5 BOLA findings

• Team A fixes vulnerabilities in avg 2 days; Team B averages 23 days

• This codebase has a 78% recurrence risk based on historical commit patterns

2. Near-Real-Time Scheduled Probes (Every 5 Minutes)

GitHub Actions cron runs as frequently as every 5 minutes. Each probe logs in as multiple users, fires crafted requests, captures response bodies, parses resource attributes (title, owner_id, description), and creates a GitHub Issue with Copilot's fix attached if a vulnerability is confirmed.

```
on:
  schedule:
    - cron: '*/5 * * * *'   # every 5 min, 24/7
  push:
    branches: [main]         # AND instantly on every deploy

# Each run: captures which data leaked, from whose account,
# full HTTP evidence — creates trackable Issue with fix attached
```

3. PR-Level Prevention — Faster Than Real-Time

Real-time monitoring catches vulnerabilities after they're in production. GitHub stops them before they're merged:

```
Developer writes vulnerable code
         ■
```

```
   Opens Pull Request
        ■
   CodeQL scans (~2 min) ■■■ flags BOLA / missing ownership check
        ■
   Copilot Autofix ■■■■■■■■■■■ writes the WHERE owner_id = $2 fix
        ■
   Required Workflow ■■■■■■■■■ runtime probe against staging fails
        ■
   PR BLOCKED — cannot merge
        ■
   Vulnerability never reaches production
        ■
   Kong never needs to handle it
```

4. Cross-Repo Org-Wide AI Analysis

Kong sees one gateway. GitHub sees every repo simultaneously. Copilot identifies the same vulnerable pattern copy-pasted across 8 microservices, or the same developer introducing BOLA in 3 different repos:

```
- name: Scan all org repos for BOLA via Security Overview API
  run: |
    gh api /orgs/$ORG/security-overview/alerts \
      --field severity=critical \
      --field alert_type=code_scanning \
      | jq '[.[] | select(.rule.id | contains("bola"))]'

    # Copilot analyzes: which repos, which teams,
    # how old each finding is, remediation velocity
```

Proposed Architecture for Cushman & Wakefield

Layer	Tool	Role	Trigger
Shift Left (Prevention)	CodeQL + Copilot Autofix	Flags BOLA in code before merge. Writes the fix automatically.	Every PR
Shift Left (Prevention)	Required Workflows	Runtime probe on staging. Blocks PR if vuln confirmed.	Every PR
Real-Time (Detection)	Azure APIM / Kong	Blocks malicious requests at the gateway layer.	Every API call (milliseconds)
Near-Real-Time (Compliance)	GitHub Actions (scheduled)	Full cross-user probes. Captures data exposed, creates Issues with fixes.	Every 5 min + on deploy
Historical AI (Intelligence)	Copilot + GitHub GraphQL	90-day trend analysis. Recurrence rates, team velocity, risk prediction.	Daily or on-demand

Org-Wide (Visibility)	Security Overview (GHAS)	Single pane of glass across all repos. CISO dashboard.	Continuous

The One-Sentence Close

"Kong and Azure APIM block the attack in milliseconds at the gateway. GitHub and Copilot prevent the vulnerability from existing in the first place, analyze 90 days of history to find systemic patterns across your whole org, and automatically write the code fix — so your security posture improves continuously, not just reactively."

Key Differentiator to Emphasize

Kong and APIM tell you **what happened in production**. GitHub + AI tell you **why it keeps happening in your codebase** and fix it at the root. That's the shift from reactive security tooling to proactive, AI-driven security engineering.