

Security team training

GitHub Advanced Security



Agenda

Setting the scene

Advanced Security features and how they fit into a secure development workflow.

Configuring access

Creating teams and applying appropriate permissions.

Reviewing alerts

Use the integrated reporting facilities to identify common issues and understand risk factors.

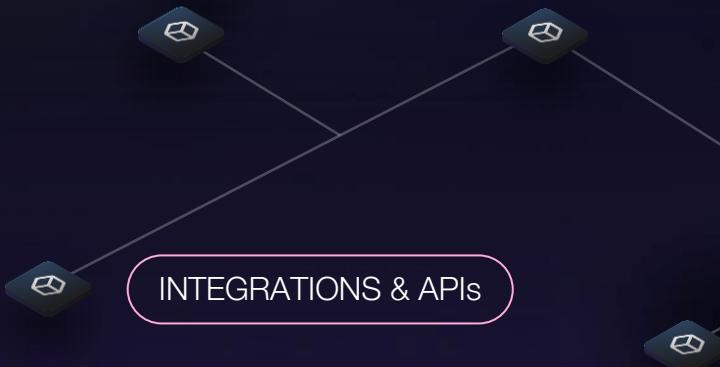
Monitoring and responding to alerts

Notifications, webhooks and APIs; building custom workflows.

Security Overview



AI-powered developer platform



GitHub Advanced Security

Feature List

Supply Chain

Know your environment

- GitHub Advisory Database
- Dependency Graph
- Dependency Insights
- Dependency Submission API
- Software Bill of Materials (SBOM)

Manage your environment

- Dependabot Alerts
- Dependabot Security Updates
- Dependabot Version Updates
- Dependency Review

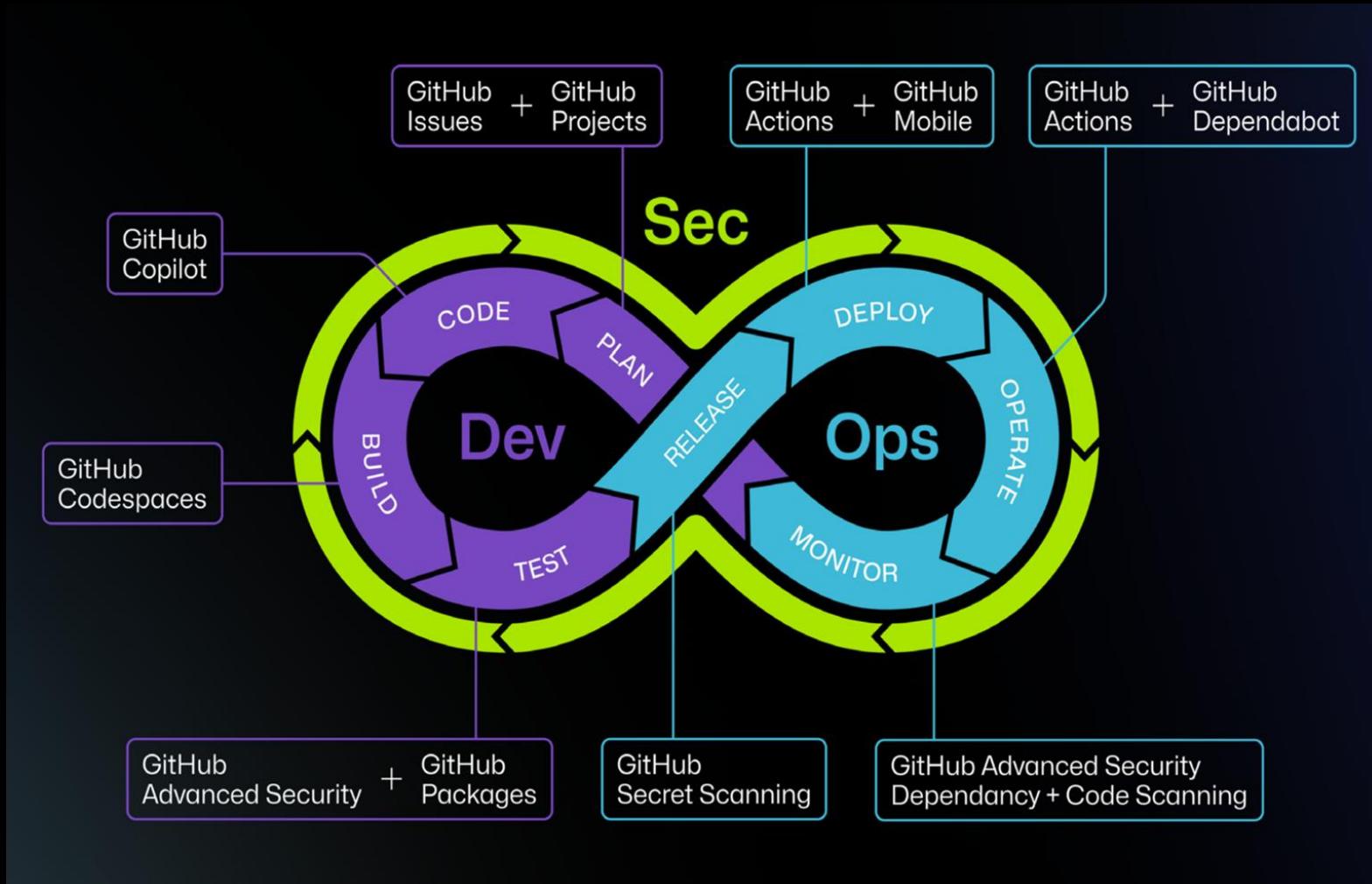
Secret Scanning

- Partner pattern and non provider pattern detection by default
- Push Protections for users and repositories
- Custom Patterns
- Regex generation with AI
- Generic Secret AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD
- Run on push, pull request, schedule or manual trigger
- SARIF output
- Third-party code scanning tool support

← Security Overview Dashboard →



Configuring Access

Security Manager Role

Members of a team with the security manager role have only the permissions required to effectively manage security for the organization.

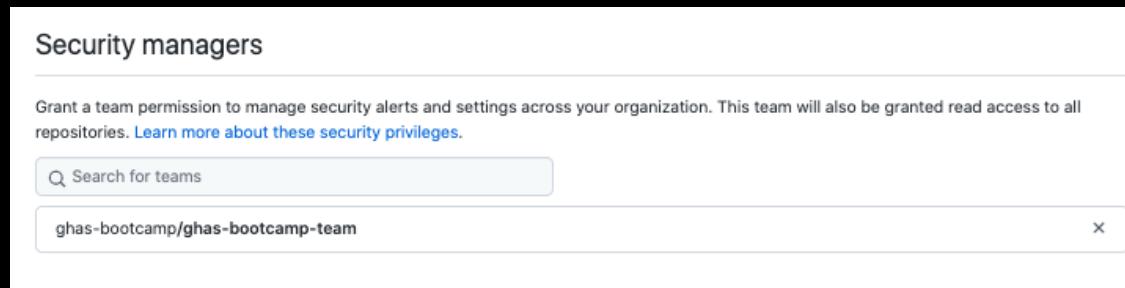
- **Read** access on all repositories in the organization, in addition to any existing repository access
- **Write** access on all security alerts in the organization
- The ability to configure security settings at the organization level
- The ability to configure security settings at the repository level
- If your organization has a security team, you can use the security manager role to give members of the team the least access they need to the organization

Security managers

Grant a team permission to manage security alerts and settings across your organization. This team will also be granted read access to all repositories. [Learn more about these security privileges.](#)

Search for teams

ghas-bootcamp/ghas-bootcamp-team X



Access Requirements for Security Features

- You can customize access to each repository in your organization by assigning granular roles, giving people access to the features and tasks they need
- Following are the default roles and the repository level actions that a person can perform with respect to the different security features

Repository action	Read	Triage	Write	Maintain	Admin
Receive Dependabot alerts for insecure dependencies in a repository	x	x	✓	✓	✓
Dismiss Dependabot alerts	x	x	✓	✓	✓
Designate additional people or teams to receive security alerts	x	x	✗	✗	✓
Create security advisories	x	x	✗	✗	✓
Manage access to GitHub Advanced Security features (see " Managing security and analysis settings for your organization ")	x	x	✗	✗	✓
Enable the dependency graph for a private repository	x	x	✗	✗	✓
View dependency reviews	✓	✓	✓	✓	✓
View code scanning alerts on pull requests	✓	✓	✓	✓	✓
List, dismiss, and delete code scanning alerts	x	x	✓	✓	✓
View and dismiss secret scanning alerts in a repository	x	x	✓	✓	✓
Resolve, revoke, or re-open secret scanning alerts	x	x	✓	✓	✓
Designate additional people or teams to receive secret scanning alerts in repositories	x	x	✗	✗	✓

Permission Flow to Access Data in Security Overview

<u>Organization member with</u>	<u>Overview dashboard view</u>	<u>Risk and alerts views</u>	<u>Coverage view</u>
admin access for one or more repositories	View data for those repositories	View data for those repositories	View data for those repositories, and enable and disable security features
write access for one or more repositories	View code scanning and Dependabot data for those repositories	View code scanning and Dependabot data for those repositories	No access for those repositories
Security alert access for one or more repositories	View all security alert data for those repositories	View all security alert data for those repositories	No access for those repositories
Custom organization role with permission to view one or more types of security alert	View allowed alert data for all repositories	View allowed alert data for all repositories in all views	No access

GHAS Policies & Settings

GHAS Security Scanning Policies

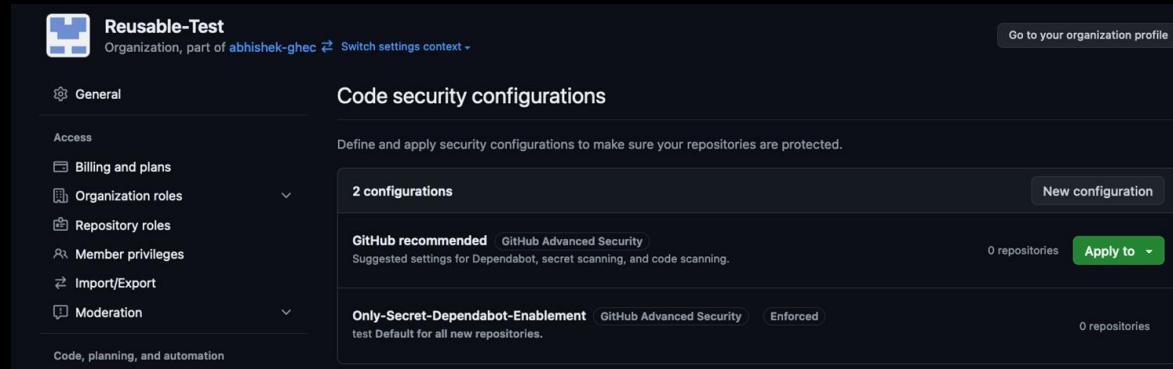
- GHAS Enablement policies can be defined at the Enterprise level or it can also be configured to delegate it to the Org and Repo Owners
- It is important to get the Policy definitions ironed out during the start of the rollout so that Security managers can have a uniform enforcement across the enterprise and there is no policy drift

The screenshot shows the GitHub Advanced Security (GHAS) settings interface. On the left, a sidebar lists various organization management options like Overview, Organizations, People, and Policies. The Policies section is currently selected. The main content area is divided into several sections:

- Code security and analysis**: A section for managing Dependabot alerts. It shows a dropdown menu set to "All organizations: Allowed".
- Dependency Insights**: A section for viewing dependency packages and security advisories. It shows a dropdown menu set to "All organizations: No policy".
- Enable or disable Dependabot alerts by repository admins**: A section for repository-level alert management. It shows a dropdown menu set to "All organizations: Allowed".
- GitHub Advanced Security policies**: A section for enabling or disabling GitHub Advanced Security on organization-owned repositories. It shows a dropdown menu set to "All repositories: Allowed".
- Repository Admins can Enable or Disable GitHub Advanced Security**: A detailed description of the policy.
- Repository Admins can Enable or Disable Secret Scanning**: A detailed description of the policy.

GHAS Code Security Configurations

- GHAS Settings can also be applied at multiple levels in GitHub
 - Enterprise, Org, Repository level
- As Security Managers and GitHub Site Owners, it is important to understand at what level do you want to enforce those settings at so that you have a sustained rollout of GHAS
- With proper planning you can have an optimal usage of your GHAS license
 - Either you enable all and roll out GHAS licenses in one go
 - Or you target your most Business critical applications and corresponding Orgs and Repos



GHAS Code Security Configurations (Continued)

- Code security configurations simplify the rollout of GitHub security products at scale by defining collections of security settings that can be applied to groups of repositories
- Apply the ‘GitHub recommended’ security configuration, which applies GitHub’s suggested settings for Dependabot, secret scanning, and code scanning or create your own custom security configurations
- Security configurations, you can also see the additional number of GitHub Advanced Security (GHAS) licenses that are required to apply a configuration, or made available by disabling GHAS features on selected repositories. This lets you understand license usage when you roll out GitHub’s code security features in your organization.
- Code Configuration lets you manage security settings based on different risk profiles and security needs

GHAS Code Security Configurations

 demo-org
Organization, part of Avocado Corp. [Switch settings context ▾](#)

[Go to your organization profile](#)

[General](#)

Access

- [Billing and plans](#)
- [Organization roles](#)
- [Repository roles](#)
- [Member privileges](#)
- [Import/Export](#)

[Moderation](#)

Code, planning, and automation

- [Repository](#)
- [Codespaces](#)
- [Planning](#)
- [Copilot](#)
- [Actions](#)
- [Webhooks](#)
- [Discussions](#)
- [Packages](#)

Code security configurations Beta Opt out

Define and apply security configurations to make sure your repositories are protected.

1 configuration [New configuration](#)

GitHub recommended Suggested settings for Dependabot, secret scanning, and code scanning. 0 repositories [Apply to ▾](#)

Apply configurations

351 GitHub Advanced Security licenses available, 150 in use by Avocado Corp.. Select repositories to view license consumption information.

[Filter](#) [🔍](#)

<input type="checkbox"/>	11 repositories	No configuration 0 licenses required
<input type="checkbox"/>	repo_3 Internal Updated last week	No configuration 0 licenses required
<input type="checkbox"/>	repo_2 Internal Updated last week	No configuration 0 licenses required

Fix and Publish Vulnerability Information

SECURITY.md

- Instructions on how to report a security vulnerability in your repository
- May reside in the repository root, docs or .github folder
- When an issue is raised in the repository, a link to your repository's security policy will be present
- After someone reports a security vulnerability in your project, you can use GitHub Security Advisories to disclose, fix, and publish information about the vulnerability.

The screenshot shows a web browser window with the GitHub security documentation. The title 'Security' is at the top, followed by a section titled 'Reporting Security Issues'. It contains instructions for reporting vulnerabilities, including email addresses for sending reports and a list of details to include. Below this is a 'Policy' section with a link to GitHub's Safe Harbor Policy.

Thanks for helping make GitHub safe for everyone.

Security

GitHub takes the security of our software products and services seriously, including all of the open source projects we host. We encourage you to report security issues to us through our GitHub organizations, such as [GitHub](#).

Even though [open source repositories are outside of the scope of our bug bounty program](#) and there are no monetary rewards, we will ensure that your finding gets passed along to the appropriate maintainers for review.

Reporting Security Issues

If you believe you have found a security vulnerability in any GitHub-owned repository, please report it to us via email. Instead, please send an email to opensource-security@github.com, s-samadi@github.com, the GitHub Security team.

Please include as much of the information listed below as you can to help us better understand and address the issue:

- The type of issue (e.g., buffer overflow, SQL injection, or cross-site scripting)
- Full paths of source file(s) related to the manifestation of the issue
- The location of the affected source code (tag/branch/commit or direct URL)
- Any special configuration required to reproduce the issue
- Step-by-step instructions to reproduce the issue
- Proof-of-concept or exploit code (if possible)
- Impact of the issue, including how an attacker might exploit the issue

This information will help us triage your report more quickly.

Policy

See [GitHub's Safe Harbor Policy](#)

Security Advisories

- Security advisories allow repository maintainers to privately discuss and fix a security vulnerability in a project.
- GitHub Security Advisories builds upon the foundation of the Common Vulnerabilities and Exposures (CVE) list. The security advisory form on GitHub is a standardized form that matches the CVE description format.

The screenshot shows a web browser window titled "Open a draft security advisory". The page contains fields for "Advisory Details" (Title, CVE identifier), "Description" (Impact, Patches, Workarounds, References), and "Affected products" (Ecosystem, Package name, Affected versions, Patched versions). The "Description" section includes rich text editor controls and placeholder text for each field.

Advisory Details

Title *

CVE identifier

Request CVE ID later

Description *

Write Preview

Impact
What kind of vulnerability is it? Who is impacted?

Patches
Has the problem been patched? What versions should users upgrade to?

Workarounds
Is there a way for users to fix or remediate the vulnerability without upgrading?

References
Are there any links users can visit to find out more?

Affected products

Ecosystem * Package name

Select an ecosystem e.g. example.js

Affected versions Patched versions

e.g. < 1.2.3 e.g. 1.2.3

+ Add another affected product

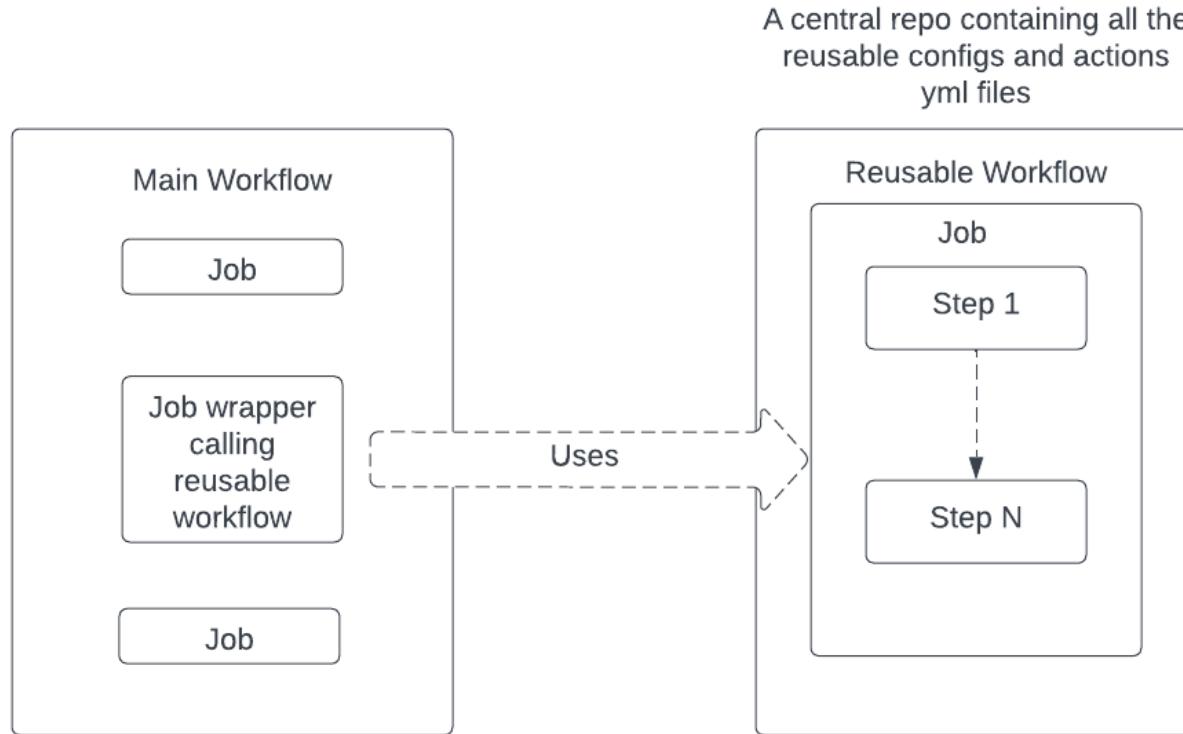
Severity

GHAS Reusability & Enforcements

Code Scanning & Dependabot Rollout & Enforcements

- As we know CodeQL and dependabot supports a configuration based (yml) execution definition in GitHub, it may be a good discussion point to see how these configurations can be streamlined and reused to stop having configuration drifts across org and repos
- There are multiple ways to achieving reusability in GitHub and its associated features
 - **Use Reusable workflows**
 - **Use Required workflows**
- Maintaining a centralized repository, with all the CodeQL yml files dependabot yml files and having the calling repo call the centralized repository or using GitHub Repo rulesets to enforce a particular configuration on the repositories under a given Org can help streamline a standardized rollout configuration

Reusable Workflow for CodeQL



Reusable Workflow for CodeQL

```
name: "CodeQL"

on:
  push:
    branches: [ "master" ]
  pull_request:
    # The branches below must be a subset of the branches above
    branches: [ "master" ]

  workflow_dispatch:

jobs:
  call-workflow-passing-data:
    uses: Reusable-Test/abhi-called-repo/.github/workflows/called-codeql-workflow.yml@main
    with:
      language: 'java'
      runner-type: 'ubuntu-latest'
      configuration-file: './.github/codeql/high-severity.yml'
```

abhi-called-repo / .github / workflows / called-codeql-workflow.yml ↗

abhi-github-staff Update called-codeql-workflow.yml

Code Blame 122 lines (103 loc) · 4.77 KB

```
1   name: Called workflow
2
3   on:
4     workflow_call:
5       inputs:
6         language:
7           required: true
8           type: string
9         runner-type:
10           required: true
11           type: string
12           configuration-file:
13             required: true
14             type: string
15
16
17   jobs:
18     analyze:
19       name: Analyze
20       # Runner size impacts CodeQL analysis time. To learn more, please see:
21       #   - https://gh.io/recommended-hardware-resources-for-running-codeql
22       #   - https://gh.io/supported-runners-and-hardware-resources
23       #   - https://gh.io/using-larger-runners
24       # Consider using larger runners for possible analysis time improvements.
25       runs-on: ${{ inputs.runner-type }}
26       timeout-minutes: ${{ (matrix.language == 'swift' && 120) || 360 }}
27       permissions:
28         # required for all workflows
```

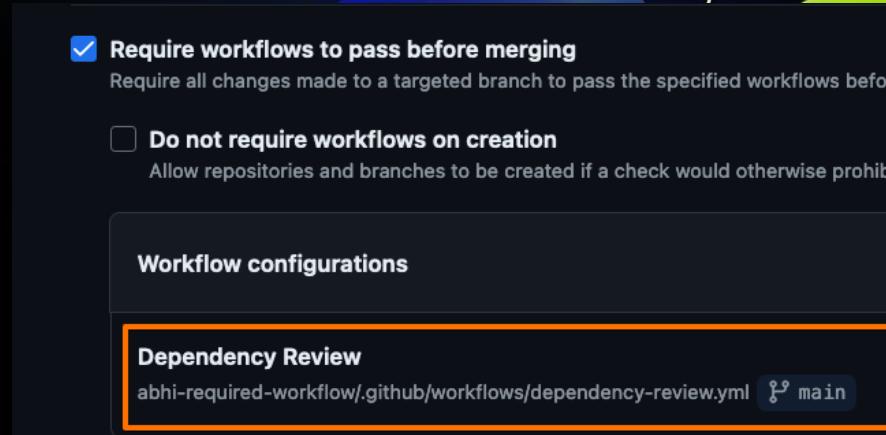
Calling Workflow

Called Reusable Workflow

Required Workflows

Another way to enforce a centralized configuration in GitHub across a set of Repositories or all repositories under an org in through the use of Required workflow (available under the repository rules in an Org or Repo level

Note: Enforcing the dependency-review.yml file in the example here will enforce dependency checks on every Pull Request



The screenshot shows the 'Required workflows' section in GitHub repository settings. It includes two options: 'Require workflows to pass before merging' (checked) and 'Do not require workflows on creation' (unchecked). Below this is a 'Workflow configurations' section with a highlighted 'Dependency Review' entry, which is linked to the file 'abhi-required-workflow/.github/workflows/dependency-review.yml'.

github-actions bot commented on Jul 2

Dependency Review

The following issues were found:

- ✗ 1 vulnerable package(s)
- ✓ 0 package(s) with incompatible licenses
- ✓ 0 package(s) with invalid SPDX license definitions
- ✓ 0 package(s) with unknown licenses.

See the Details below.

Vulnerabilities

dvcsharp-core-api.csproj

Name	Version	Vulnerability	Severity
NuGetCommandLine	6.7.0	NuGet Client Security Feature Bypass Vulnerability	critical

OpenSSF Scorecard

Package	Version	Score	Details
nuget/NuGetCommandLine	6.7.0	6.8	► Details

Scanned Manifest Files

► dvcsharp-core-api.csproj

⌚

Some checks were not successful
1 failing and 3 successful checks

Hide all checks

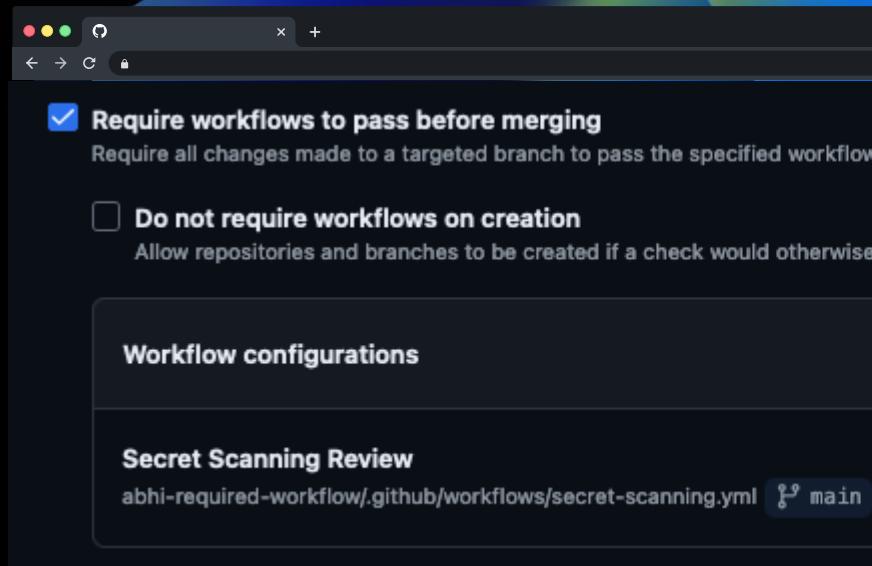
✗ Dependency Review / dependency-review (pull_request) Failing after 4s	Details
✓ CodeQL / call-workflow-passing-data / Analyze (javascript-typescript, none) (pull_request) S...	Details
✓ CodeQL / call-workflow-passing-data / Analyze (csharp, none) (pull_request) Successful in 3m	Details
✓ Code scanning results / CodeQL Successful in 26s — No new alerts in code changed by this pull ...	Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

Dependency Review check
enforced at the Pull Request Level
based on the required workflow
definition in the Repo Ruleset

Required Workflows

1. One can also enforce **Secret Scanning** checks at the Pull Request level by adding a secret-scanning.yml file (using secret scanning review action)
2. Once secret scanning yml is defined in the repo rules as part of the config "Require workflows to pass before merging", the action will check if any new secret is leaked in the repository from the feature branches
3. Once you enforce this definition on all/selected repos under the organization, any net new secrets leaked into the repo, the PR will be blocked from merging in case of new vulnerability introduction



Workflow file for this run

.github/workflows/secret-scanning.yml at 1da9c50

```
1 name: 'Secret Scanning Review'
2 on: [pull_request]
3
4 jobs:
5   secret-scanning-review:
6     runs-on: ubuntu-latest
7     steps:
8       - name: 'Secret Scanning Review Action'
9         uses: advanced-security/secret-scanning-review-action@v1
10        with:
11          token: ${{ secrets.SECRET_SCAN_REVIEW_GITHUB_TOKEN }}
12          fail-on-alert: true
13          fail-on-alert-exclude-closed: true
```

The screenshot shows a GitHub pull request page with a summary of secret scanning findings. At the top, it says "PR#10 SECRET SCANNING REVIEW SUMMARY". It indicates that 2 secrets were found across 2 locations. Below this is a table with columns: Status, Secret Alert, Secret Type, State, Resolution, Push Bypass, and Commit. Two rows are listed:

Status	Secret Alert	Secret Type	State	Resolution	Push Bypass	Commit
🔴	⚠️ 8	my-token	open	✗	False	1da9c50
🔴	⚠️ 2	test-pattern-1	open	✗	False	1da9c50

Below the table, there's a section titled "Some checks were not successful" with a count of 3 failing and 3 successful checks. It lists several check results, including:

- Dependency Review / dependency-review (pull_request) Failing after 5s
- Secret Scanning Review / secret-scanning-review (pull_request) Failing after 29s
- CodeQL / Analyze (javascript-typescript, none) (pull_request) Successful in 1m
- CodeQL / Analyze (csharp, none) (pull_request) Successful in 3m
- CodeQL / compliance (pull_request) Failing after 1s

Typical secret-scanning.yml file with the policy definitions ([Actions Link](#))

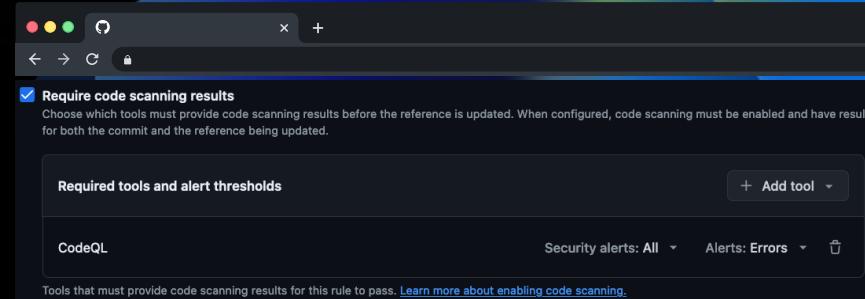
A PR level summary will be created with the list of all new secrets identified along with the PR level checks (failure in this case)

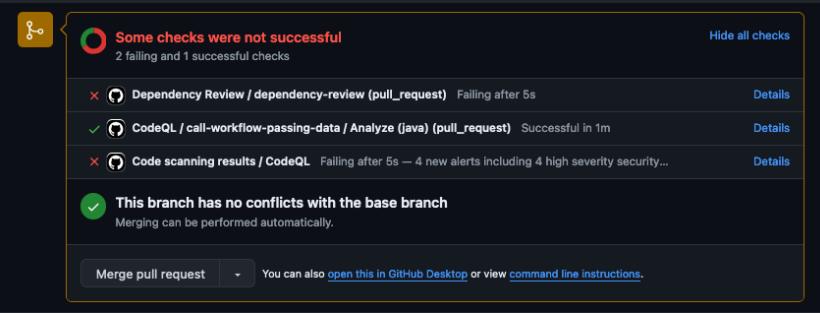
Required Workflows

Another way of **enforcing CodeQL** scans on every Pull Request is by configuring the : **Required code scanning results**

In the Repository ruleset configuration page in the GitHub UI

Note: As a Security Architect, one can define the permissible alerts threshold in the configuration and when a developer merging mode from their respective feature branches by creating a Pull Request to a protected branch, the configuration would check for any new vulnerability introduced. If new vulnerabilities are detected with the mentioned criticality, the PR would be blocked





CodeQL vulnerability threshold enforced at the Pull Request level through the Repo Rulesets

A screenshot of a GitHub pull request interface titled "testing data #4". It shows a "Checks" tab with a failure for "GitHub Advanced Security / CodeQL". The "Code scanning results" section shows "4 new alerts including 4 high severity security vulnerabilities". One alert is expanded, detailing a "Check failure on line 48" due to "Uncontrolled data used in path expression" with a "High" severity. Another alert is partially visible below it.

The net new vulnerability details available within the Pull Request

Best Practices and Considerations Checklist



- 💡 Establish a branching strategy
- 💡 Manage repositories with custom properties
- 💡 Start with managing an inventory of the repositories by business criticality/language
- 💡 Use Security Configurations where possible
- 💡 Create Repository Rulesets for supplemental enforcement
- 💡 Consider your desired access permissions
- 💡 Create and monitor an exception process

Notifications and Alerts

Security Manager role

Members of a team with the security manager role have only the permissions required to effectively manage security for the organization.

- **Read** access on all repositories in the organization, in addition to any existing repository access
- **Write** access on all security alerts in the organization
- The ability to configure security settings at the organization level
- The ability to configure security settings at the repository level

Security managers

Grant a team permission to manage security alerts and settings across your organization. This team can manage security for up to 100 repositories. [Learn more about these security privileges.](#)

Search for teams

ghas-bootcamp/ghas-bootcamp-team

Default Permissions



Repository action	Read	Triage	Write	Maintain	Admin
Receive Dependabot alerts for insecure dependencies in a repository	✗	✗	✓	✓	✓
Dismiss Dependabot alerts	✗	✗	✓	✓	✓
Designate additional people or teams to receive security alerts	✗	✗	✗	✗	✓
Create security advisories	✗	✗	✗	✗	✓
Manage access to GitHub Advanced Security features (see " Managing security and analysis settings for your organization ")	✗	✗	✗	✗	✓
Enable the dependency graph for a private repository	✗	✗	✗	✗	✓
View dependency reviews	✓	✓	✓	✓	✓
View code scanning alerts on pull requests	✓	✓	✓	✓	✓
List, dismiss, and delete code scanning alerts	✗	✗	✓	✓	✓
View and dismiss secret scanning alerts in a repository	✗	✗	✓	✓	✓
Resolve, revoke, or re-open secret scanning alerts	✗	✗	✓	✓	✓
Designate additional people or teams to receive secret scanning alerts in repositories	✗	✗	✗	✗	✓

Notifications

Secret Scanning

- Incremental
- Historical
- Watch repository

Dependabot

- Watch repository
- Communication channels
- Frequency

Code Scanning

- No traditional notifications
- Can be achieved via webhooks

Custom repository roles for security features

- When you create a custom repository role, you start by choosing an inherited role from a set of pre-defined options.
- The inherited role determines the initial set of permissions included in the custom role.
- Then, you can further customize the role by choosing additional permissions to give the role.

Repository roles / Create a role

Beta

Roles are used to grant access and permissions for teams and members. Begin by choosing a role to inherit and adding permissions to create a role that fits your needs. [Learn more](#)

Name

Test role

Description

What is this role all about?

A short description who this role is for or what permissions it grants

Choose a role to inherit

All custom roles must inherit the permissions of a default role.

 Read  Triage  Write  Maintain

Add Permissions

Add permissions to create a role that fits your needs.

<input type="checkbox"/>	Manage webhooks	Repository
<input type="checkbox"/>	Manage deploy keys	Repository
<input type="checkbox"/>	View code scanning results	Security
<input type="checkbox"/>	Dismiss or reopen code scanning results	Security
<input type="checkbox"/>	Delete code scanning results	Security
<input type="checkbox"/>	View secret scanning results	Security
<input type="checkbox"/>	Dismiss or reopen secret scanning results	Security

Configuring access - Demo

Reviewing Alerts

Hands-on demo

Integrations - GHAS API and Webhooks

APIs

REST

/code-scanning (Enterprise, Organization &
Repository)

/dependabot (Enterprise, Organization &
Repository)

/dependency-graph (Repository)

/secret-scanning (Enterprise,
Organization & repository)

GraphQL

DependabotUpdate

SecurityVulnerability

SecurityAdvisory

Preview:
DependencyGraph

Webhooks

- 
- 1 *code_scanning_alert* (Repository, Organization, App)
 - 2 *dependabot_alert* (Repository, Organization, App)
 - 3 *secret_scanning_alert* (Repository, Organization, App)
 - 4 *secret_scanning_alert_location* (Repository, Organization, App)
 - 5 *security_advisory* (App)
 - 6 *security_and_analysis* (Repository, Organization, App)

Audit log

Events Settings

Audit log

Filters Export Git Events Export

Clear current search query

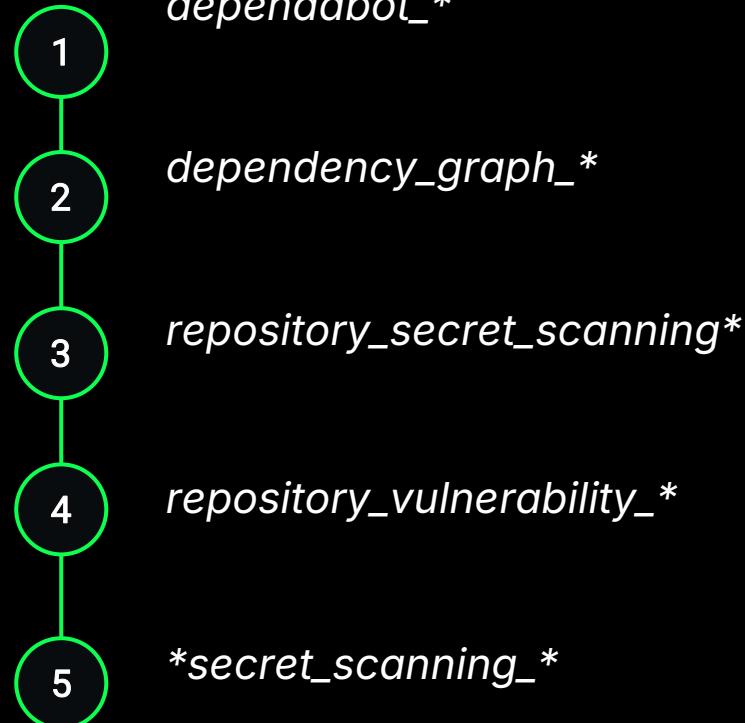
Events matching search query

abhi-github-staff - **secret_scanning_push_protection.bypass**
Bypassed the push protection for a secret as false positive Reusable-Test/JuiceShop:alert#45
Unknown location | Unknown IP address | 19 hours ago | ...

abhi-github-staff - **secret_scanning_push_protection.bypass**
Bypassed the push protection for a secret as used in tests Reusable-Test/JuiceShop:alert#44
Unknown location | Unknown IP address | 19 hours ago | ...

Newer Older

ProTip! Exclude events created by you with [-actor:abhi-github-staff](#)





Monitoring and responding to alerts

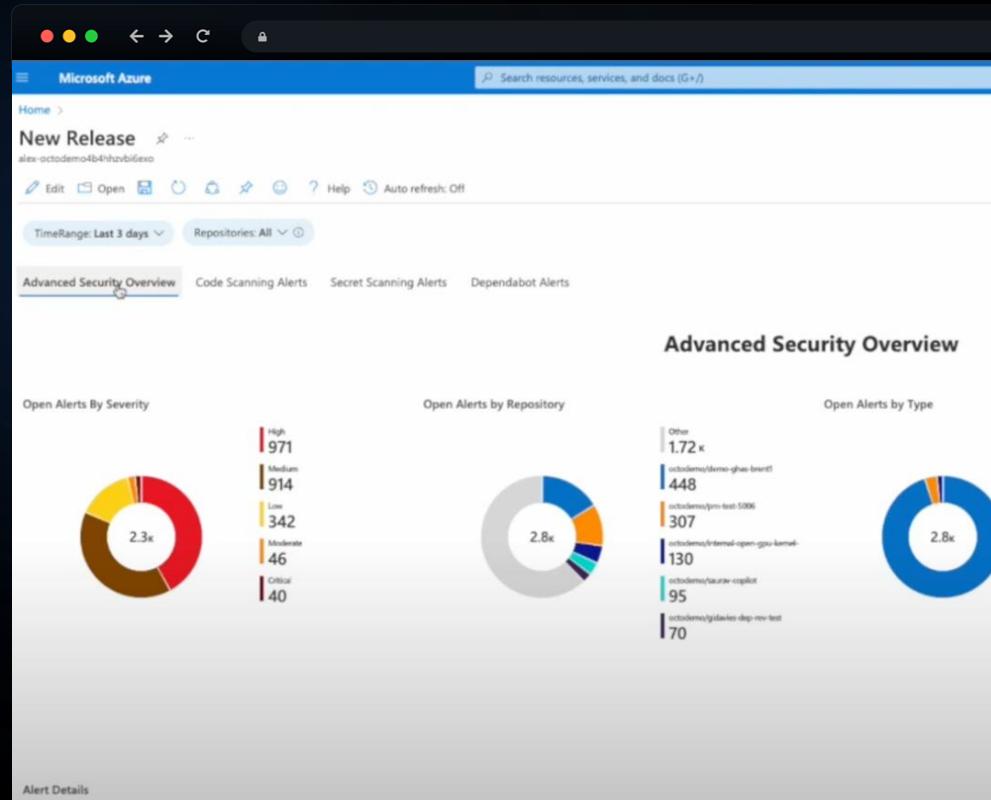
Hands-on demo



3rd Party Integrations

SIEM Integrations

- ▷ Splunk
- ▷ Azure Sentinel
- ▷ Data Dog
- ▷ Sumo Logic
- ▷ Elastic



JIRA Integration with GHAS Vulnerabilities

JIRA now natively supports ingesting GHAS vulnerabilities into JIRA and subsequently creating issues in JIRA in an automated way

<https://support.atlassian.com/jira-cloud-administration/docs/integrate-github-for-security/>

<https://www.youtube.com/watch?v=QA-1gjzSuYc>

Integrate GitHub Advanced Security with Jira

i These instructions are for connecting GitHub Cloud or GitHub Enterprise Cloud to Jira. [Show me how to connect GitHub Enterprise Server](#)

The security feature in Jira allows you to view, triage, and track security vulnerabilities from GitHub Advanced Security. To get this feature working, you'll need to:

1. Install the GitHub for Jira app.
2. Connect a GitHub organization.
3. Add GitHub Advanced Security to your Jira project.
4. Connect security containers to your project.

Before you begin

i To install and set up the GitHub for Jira app, you need:

- Site administrator permission for your Jira site.
- Organization owner permission for your GitHub organization.

For some organizations, the task of integrating GitHub Advanced Security might involve multiple team members:

- A Jira site admin will install the GitHub for Jira app.
- A GitHub organization owner will connect a GitHub organization to your Jira site.
- A Jira project admin will add GitHub Advanced Security to a project and connect security containers.

3rd Party Scanner Integration and Data Ingestion

Code Blame 43 lines (37 loc) · 1.09 KB

```
1 name: scan with KICS and upload SARIF
2
3 on:
4   push:
5     branches: [master]
6
7 jobs:
8   kics-job:
9     runs-on: ubuntu-latest
10    name: kics-action
11    strategy:
12      fail-fast: false
13    steps:
14      - name: Checkout repo
15        uses: actions/checkout@v3
16      - name: Mkdir results-dir
17        # make sure results dir is created
18        run: mkdir -p results-dir
19
20      - name: Run KICS Scan with SARIF result
21        uses: checkmarx/kics-github-action@v2.1.0
22        with:
23          path: 'terraform'
24          output_path: results-dir
25          platform_type: terraform
26          output_formats: 'json,sarif'
27          ignore_on_exit: results
28
29      - name: Show results
30        run:
31          cat results-dir/results.sarif
32          cat results-dir/results.json
33
34      - name: Archive code coverage results
35        uses: actions/upload-artifact@v4
36        with:
37          name: result
38          path: results-dir/results.sarif
39
40      - name: Upload SARIF file
41        uses: github/codeql-action/upload-sarif@v1
42        with:
43          sarif_file: results-dir/results.sarif
```



GitHub Code Scanning allows 3rd party scanner data ingestion

Output data format should be SARIF

If running the scan (ex. IaC) in GitHub Actions, use **upload-sarif**

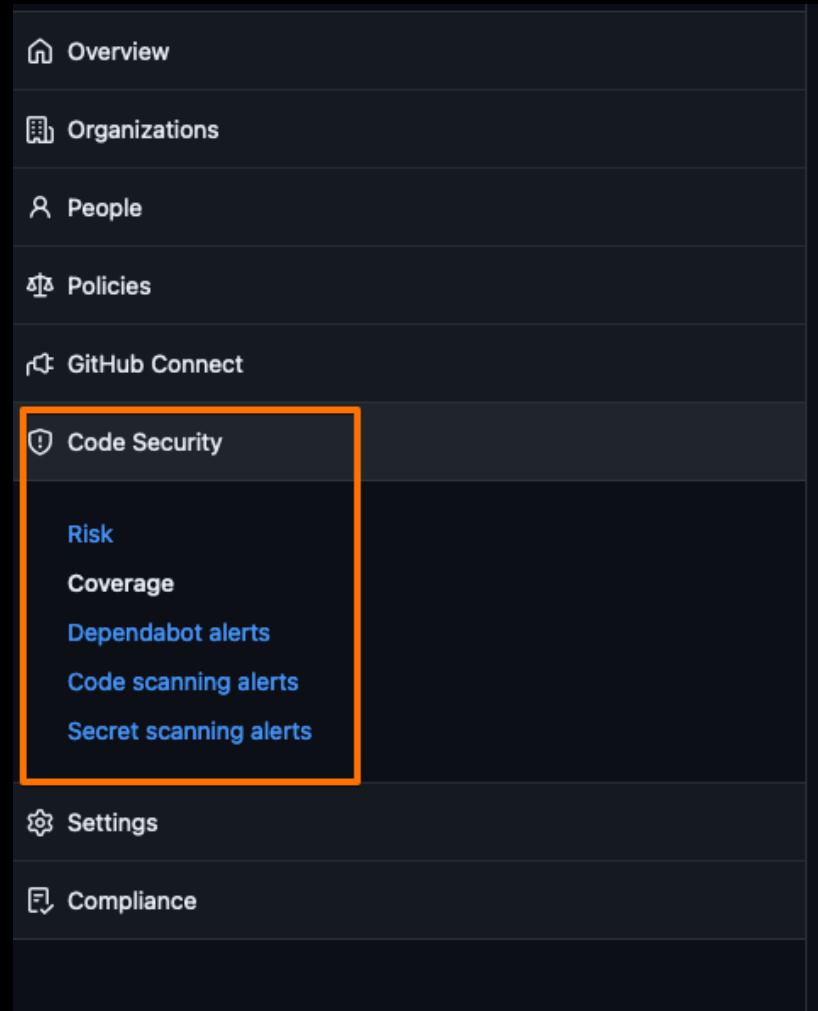
Upload can also be done through REST API

Vulnerabilities can be viewed in Security Dashboard in GitHub

Security Overview

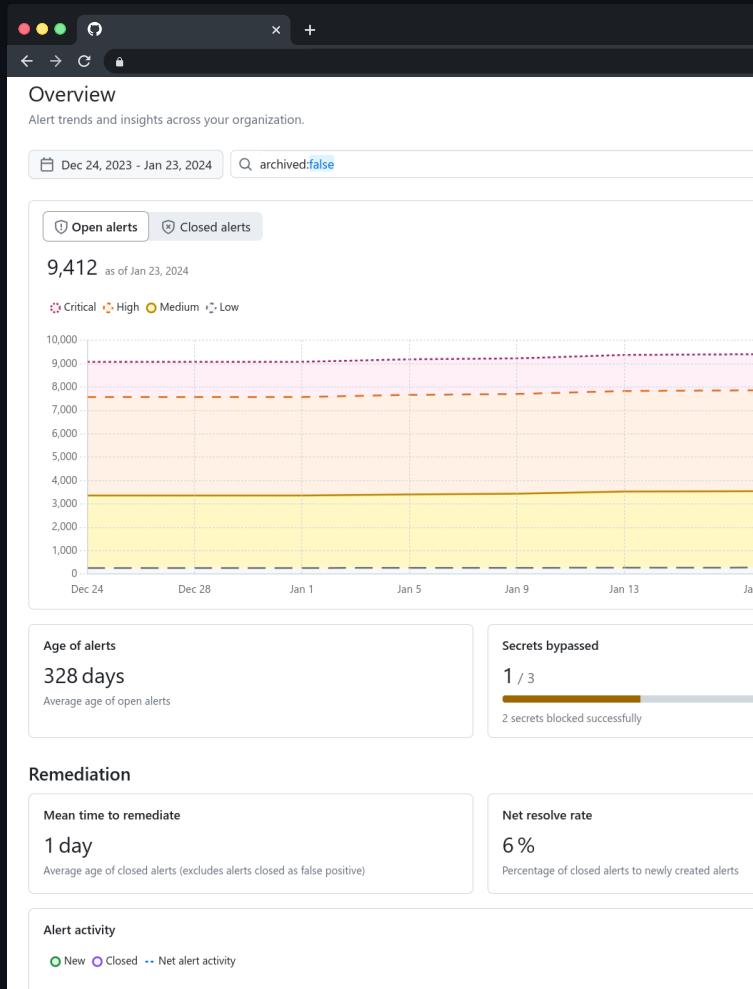
Enterprise Level View

- You can find security overview on the Code Security tab for your enterprise. Each page displays aggregated and repository-specific security information for your enterprise.
- In the enterprise-level security overview, you can see data for all organizations where you are an organization owner or security manager.
- As a security team member, you cannot use the enterprise-level security overview to enable and disable security features



Org Level View

- Provides high-level summaries of the security landscape of an organization or enterprise and makes it easy to identify repositories that require intervention
- Shows metrics on default branches
- At organisation level, 3 summary views:
 - Overview
 - Coverage
 - Risk
- Show data only for high confidence alerts. Code scanning alerts from third-party tools, and secret scanning alerts for ignored directories and non-provider alerts are all omitted from these views



Repo Level View

- At the repo level, drill down at the vulnerabilities across the 3 pillars of GHAS
 - Dependabot
 - Code Scanning
 - Secret Scanning
 - High Confidence
 - Other

The screenshot shows the GitHub Dependabot alerts interface. On the left, there's a sidebar with navigation links: Overview, Reporting, Policy, and Advisories. Below that is a Requests section and a Push protection bypass link. The main area is titled "Dependabot alerts" and features a "Auto-triage your alerts" section with a brief description and a "Learn more about auto-triage" link. A search bar at the top right contains the query "is:open". The main content area lists vulnerabilities under "Vulnerability alerts". It shows a summary: 4 Open (with 0 Closed). Below this is a list of four vulnerabilities:

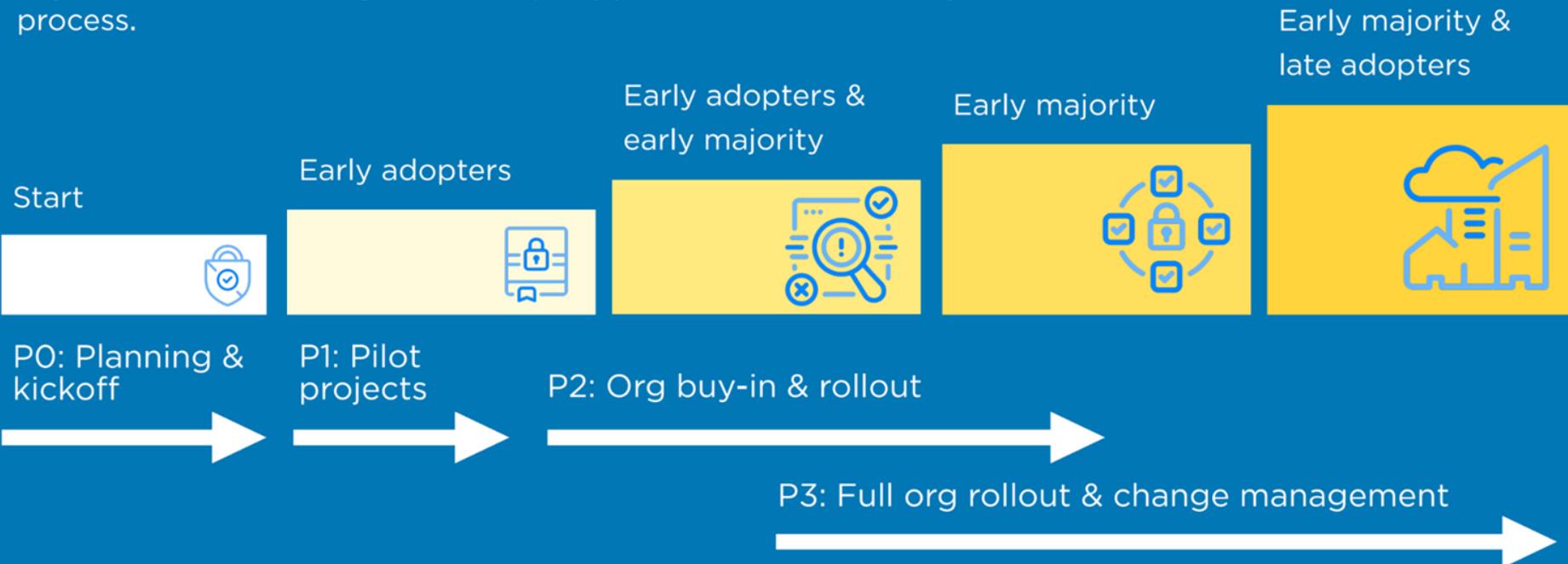
- Improper Handling of Exceptional Conditions in Newtonsoft.Json (High)
- Microsoft ASP.NET Core project templates vulnerable to denial of service
- Moderate severity vulnerability that affects Microsoft.AspNetCore.All, Microsoft.AspNetCore.Server.Kestrel.Core (Moderate)
- Moderate severity vulnerability that affects Microsoft.AspNetCore.All, Microsoft.AspNetCore.Server.Kestrel.Core, Microsoft.AspNetCore.Server.Kestrel.Transport.Libuv (Moderate)

Each item in the list includes a checkbox, a shield icon, the vulnerability title, its severity, a timestamp ("#3 opened 3 weeks ago" or "#2 opened 3 weeks ago"), the detected package ("Detected in Newtonsoft.Json (NuGet) · dvcsharp-core-api.csproj" or "Detected in System.IdentityModel.Tokens.Jwt (NuGet) · dvcsharp-core-api.csproj"), and the affected component.

Scaling GitHub Advanced Security

GHAS phased rollout & deployment

Throughout the phases of the GHAS rollout, groups within your organization can be targeted to help support the rollout and buy-in process.



Resources/Appendix

Docs/Resources

- [Whitepaper - keys to a successful rollout](#)
- [Adopting and scaling GHAS in your company](#)
- [GHAS Enablement automation](#)
- [GitHub Professional Services](#)
- [Advanced-security repo - a collection of tools for managing GHAS](#)
- [GitHub Skills - a GitHub learning platform](#)
- [GitHub Advanced Security learning path](#)
- [Code scanning API](#)
- [Secret scanning API](#)
- [CodeQL Repository](#)



Thank you