

Developer Training

GitHub Advanced Security



Agenda

Introduction

Supply Chain Security

Secret Scanning

Code Scanning

Security Overview

The state of security today

DevSecOps Today

Security Risks

Applications are the #1 attack vector.

80% of breaches occur through web application exploits.

Source: [Verizon Data Breach Investigations Reports 2023](#)

Stagnant Progress

65% of vulnerabilities still exist after 3 months.

Only 33% of breaches are discovered by an organization's teams or tools.

Source: [Veracode State of Security Report 2023](#)

Force Multiplier

Organizations believe AI offers greater ROI

84% of executives plan to prioritize generative AI cybersecurity solutions over conventional cybersecurity solutions.

Source: [IBM CEO's Guide to Generative AI, 2023](#)

Looking to the
future...

**Our attack
surface is
growing at an
unprecedented
rate**

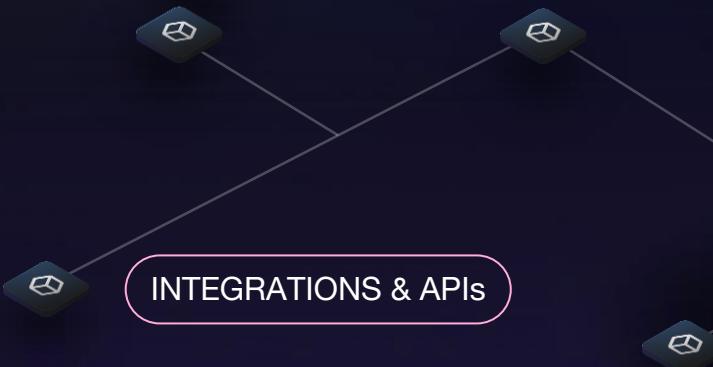
We live in a world now fully consumed by software. Every organization is a software organization and must learn how to thrive and innovate digitally.

700M

More applications will be written in the next 5 years that's more than the last 40 years combined



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 Copilot
- Generic Secrets AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD - SARIF output
- Repo rulesets
- Autofix AI
- Third-party code scanning tool support
- Security Campaigns

← Security Overview Dashboard →

Licence Feature List

What's included in each licence

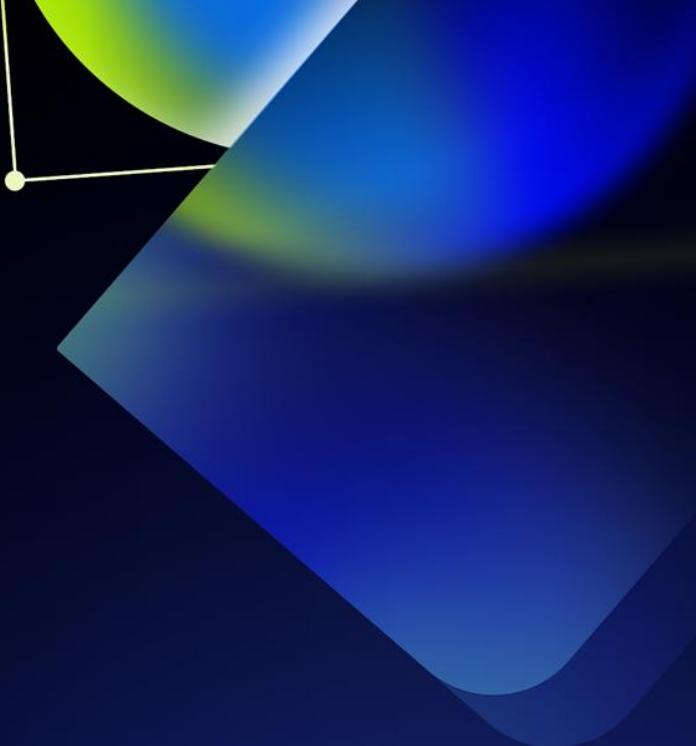
GHE Licence

- GitHub Advisory Database
- Dependency Graph
- Dependency Insights
- Dependency Submission API
- Software Bill of Materials (SBOM) generation
- Dependabot Alerts
- Dependabot Security Updates
- Dependabot Version Updates

GHAS Licence

- Dependabot auto triage rules
- Dependency Review
- All of the Secret Scanning capabilities including Copilot features
- All of the Code Scanning capabilities including Copilot features (not including runners)

Supply Chain Security



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 Copilot
- Generic Secrets AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD - SARIF output
- Repo rulesets
- Autofix AI
- Third-party code scanning tool support
- Security Campaigns

← Security Overview Dashboard →

Know Your Environment

Dependency Graph

- Summary of the manifest and lock files stored in a repository
- Includes information on your direct dependencies
- Each dependency includes the license information* and vulnerability severity
- Automatically updated when:
 - A commit is pushed that changes a supported manifest or lock file in a default branch
 - A change is pushed to a repository of one of your dependencies
- Repository and Organisation views

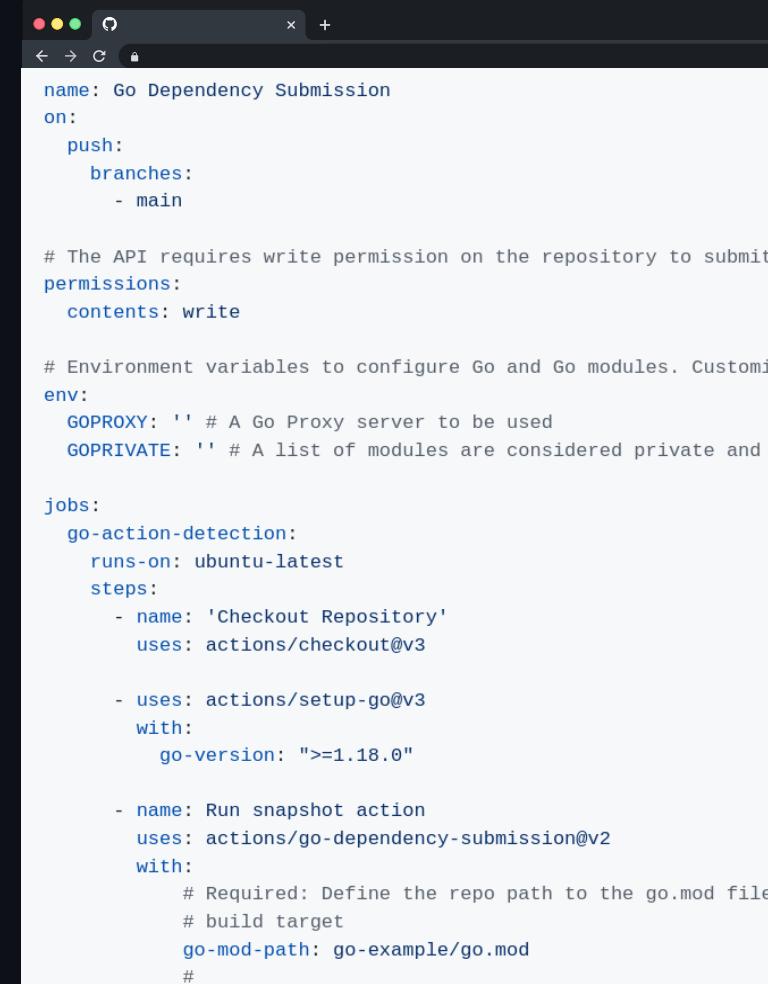
The screenshot shows a web application interface for managing dependencies. At the top, there's a navigation bar with links for Actions, Projects, Wiki, Security, Insights (which is currently selected), and Settings. On the left, a sidebar menu lists various sections: Pulse, Contributors, Community, Traffic, Commits, Code frequency, Dependency graph (which is highlighted in red), Network, Forks, and People. The main content area is titled "Dependency graph". It displays a list of dependencies with the following details:

Dependency	Version	Detected	File	License
asn1crypto	0.24.0	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	MIT
certifi	2020.6.20	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	MPL-2.0
chardet	3.0.4	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	LGPL-2.1
click	7.1.2	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	BSD-2-Clause
cryptography	2.6.1	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	Apache-2.0
django-two-factor-auth	1.12	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	MIT
entrypoints	0.3	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	MIT
flask	1.1.2	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	BSD-2-Clause
flask-cors	3.0.8	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	MIT
idna	2.10	Automatically on Nov 22, 2023	(pip) auth-ext/requirements.txt	MIT

Below the dependency list, there are tabs for Dependencies, Dependents, and Dependabot, and a search bar labeled "Search all dependencies".

Dependency Submission API

- REST API to submit dependencies for repositories
- The dependency graph shows any dependencies you submit using the API in addition to any dependencies that are identified from manifest or lock files in the repository
- Submitted dependencies will receive Dependabot alerts and Dependabot security updates for any known vulnerabilities
- Submitted in the form of a snapshot which is a set of dependencies associated with a commit SHA and other metadata, that reflects the current state of your repository for a commit.



```
name: Go Dependency Submission
on:
  push:
    branches:
      - main

# The API requires write permission on the repository to submit
permissions:
  contents: write

# Environment variables to configure Go and Go modules. Custom
env:
  GOPROXY: '' # A Go Proxy server to be used
  GOPRIVATE: '' # A list of modules are considered private and

jobs:
  go-action-detection:
    runs-on: ubuntu-latest
    steps:
      - name: 'Checkout Repository'
        uses: actions/checkout@v3

      - uses: actions/setup-go@v3
        with:
          go-version: ">=1.18.0"

      - name: Run snapshot action
        uses: actions/go-dependency-submission@v2
        with:
          # Required: Define the repo path to the go.mod file
          # build target
          go-mod-path: go-example/go.mod
          #
```

Automatic Dependency Submission

- For some ecosystems, resolution of transitive dependencies occurs at build-time
- Identifies and submits direct and transitive dependencies via the dependency submission API
- Only for Maven
- Requires GitHub Actions

Dependency	Version	Detected by	Date	Link
ch.qos.logback:logback-classic	1.2.3	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
ch.qos.logback:logback-core	1.2.3	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
com.carrotsearch.thirdparty:simple-xml-safe	2.7.1	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
com.fasterxml.jackson.core:jackson-annotations	2.11.2	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
com.fasterxml.jackson.core:jackson-core	2.11.2	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
com.fasterxml.jackson.core:jackson-databind	2.11.2	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
com.fasterxml.jackson.datatype:jackson-datatype-jdk8	2.11.2	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details
com.fasterxml.jackson.datatype:jackson-datatype-jsr310	2.11.2	Detected by Automatic Dependency Submission	on Aug 05, 2024 (Maven)	View Details

Software Bill of Materials (SBOM)

- Formal, machine-readable inventory of project dependencies
 - Includes versions, package identifiers, licenses
- Benefits:
 - Reduces supply chain risks
 - Provides transparency of repository dependencies
 - Early identification of vulnerabilities
 - Insights into license compliance, security, and quality issues
 - Helps comply with data protection standards
- Generation Methods:
 - GitHub UI
 - GitHub Actions (attached as a workflow artifact)
 - REST API
- Uses SPDX industry standard format

GitHub Advisory Database

<https://github.com/advisories>

Each advisory in the GitHub Advisory Database is for a vulnerability in open source projects or for malicious open source software

We add advisories to the GitHub Advisory Database from the following sources:

- Security advisories reported on GitHub
- The National Vulnerability database
- The npm Security advisories database
- The FriendsOfPHP database
- The Go Vulncheck database
- The Python Packaging Advisory database
- The Ruby Advisory database
- The RustSec Advisory database
- Community contributions. For more information, see <https://github.com/github/advisory-database/pulls>.

GitHub Advisory Database

<https://github.com/advisories>

The GitHub Advisory Database groups vulnerabilities into three categories:

- GitHub-reviewed advisories
- Unreviewed advisories
- Malware Advisories

The screenshot shows the GitHub Advisory Database interface. At the top, it displays "GitHub reviewed advisories" with a total count of 15,869. Below this is a table listing various ecosystems and their counts: Composer (2,539), Erlang (24), GitHub Actions (15), Go (1,387), Maven (4,443), npm (3,275), NuGet (558), pip (2,307), Pub (8), RubyGems (782), Rust (693), and Swift (33). A section for "Unreviewed advisories" shows a total of 205,633. The bottom of the page features links for CC-BY-4.0 License, Language support, and About GitHub Advisory Database.

Ecosystem	Count
Composer	2,539
Erlang	24
GitHub Actions	15
Go	1,387
Maven	4,443
npm	3,275
NuGet	558
pip	2,307
Pub	8
RubyGems	782
Rust	693
Swift	33

[CC-BY-4.0 License](#) [Language support](#) [About GitHub Advisory Database](#)



Q & A

Manage Your Environment

Dependabot Alerts

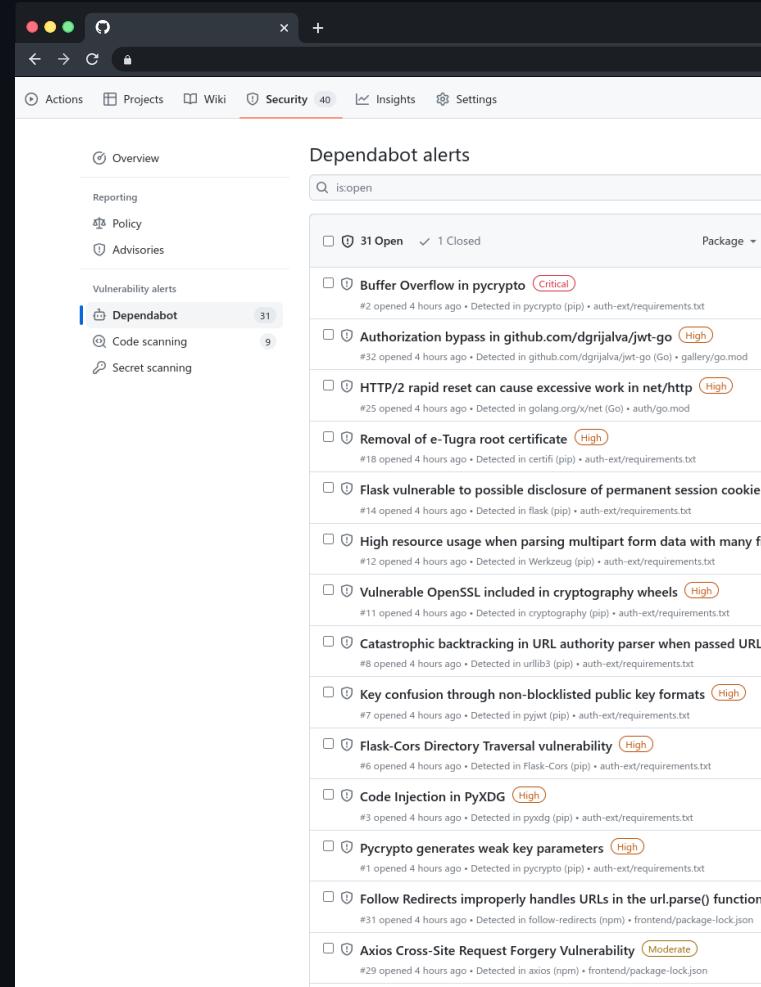
- A Dependabot alert is generated when a vulnerable dependency or malware is detected in a repository on the default branch
- Achieved by comparing a repository's dependency graph with the entries in the GitHub Advisory Database
- Admins can enable Dependabot alerts at the repository, organisation or enterprise

The screenshot shows the GitHub Dependabot alerts interface. At the top, there are tabs for Actions, Projects, Wiki, Security (40), Insights, and Settings. The Security tab is selected. On the left, there's a sidebar with Overview, Reporting, Policy, and Advisors sections. Below that is a Vulnerability alerts section with three items: Dependabot (31), Code scanning (9), and Secret scanning (0). The main area is titled "Dependabot alerts" and contains a search bar with "is:open". It lists 31 open vulnerabilities, each with a shield icon, a title, and a severity level (e.g., Critical, High, Moderate, Low). The first few items include:

- Buffer Overflow in pycrypto (Critical)
- Authorization bypass in github.com/dgrijalva/jwt-go (High)
- HTTP/2 rapid reset can cause excessive work in net/http (High)
- Removal of e-Tugra root certificate (High)
- Flask vulnerable to possible disclosure of permanent session cookie (High)
- High resource usage when parsing multipart form data with many (High)
- Vulnerable OpenSSL included in cryptography wheels (High)
- Catastrophic backtracking in URL authority parser when passed URL (High)
- Key confusion through non-blocklisted public key formats (High)
- Flask-Cors Directory Traversal vulnerability (High)
- Code Injection in PyXDG (High)
- Pycrypto generates weak key parameters (High)
- Follow Redirects improperly handles URLs in the url.parse() function (Moderate)
- Axios Cross-Site Request Forgery Vulnerability (Moderate)

Dependabot Alerts

- After enablement a scan is triggered on the default branch when
 - A new advisory is added to the GitHub Advisory Database
 - The dependency graph for a repository changes.
- Users with write, maintain or admin privileges can view Dependabot alerts once enabled.



Dependabot Notifications

- When a new Dependabot alert is detected, GitHub notifies all users with access to Dependabot alerts for the repository according to their notification preferences
- You will receive alerts if you are:
 - watching the repository
 - have enabled notifications for security alerts or for all the activity on the repository,
 - and are not ignoring the repository

If you are concerned about receiving too many notifications for Dependabot alerts, we recommend you opt into the weekly email digest, or turn off notifications while keeping Dependabot alerts enabled.

Dependabot alerts: New vulnerabilities

When you're given access to [Dependabot alerts](#) automatically receive notifications when a new vulnerability is found in one of your dependencies.

[Notify me: on GitHub, Email, CLI](#)

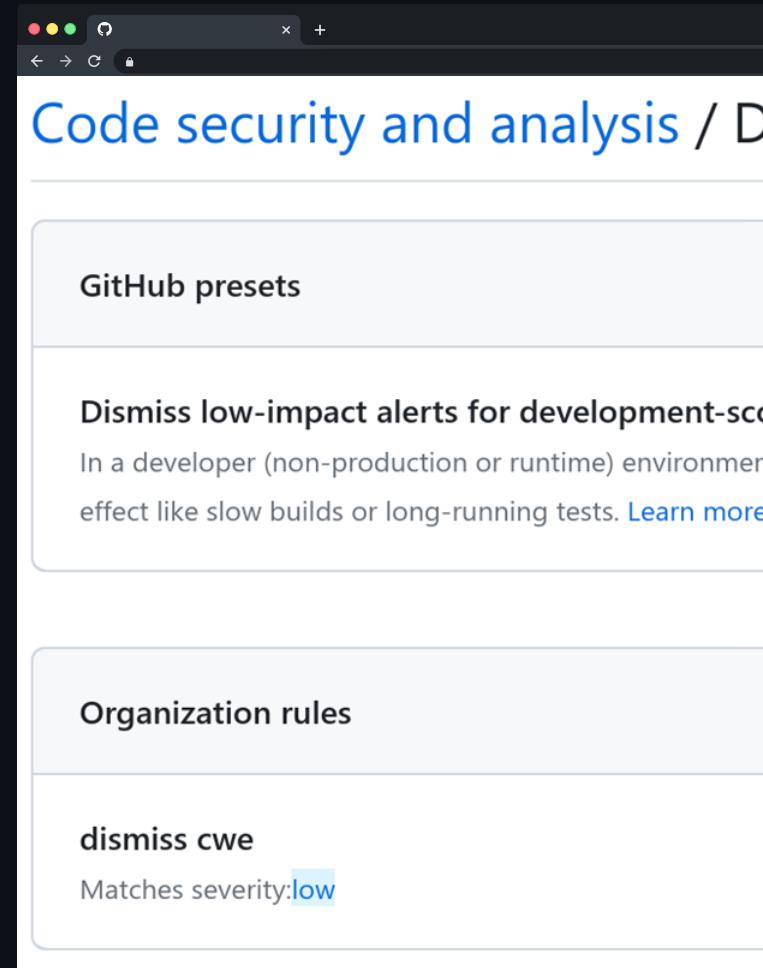
Email weekly digest

Email a weekly summary summarizing alerts for up to 10 of your repositories.

[Send weekly ▾](#)

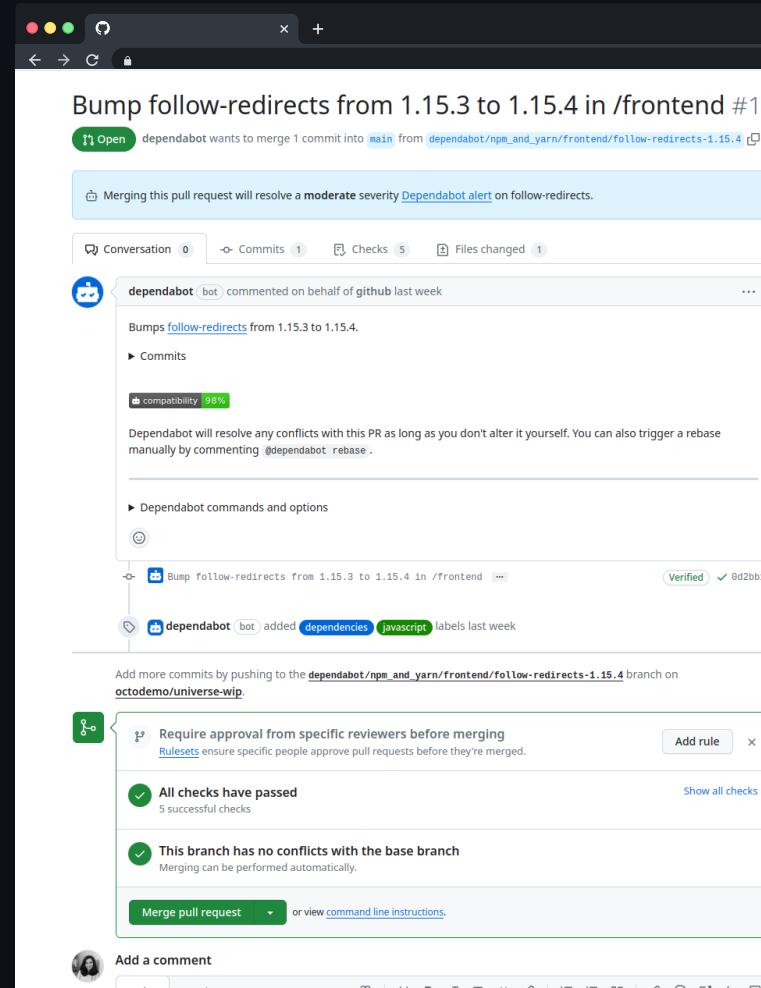
Dependabot auto-triage rules

- Instruct dependabot to automatically triage Dependabot alerts based on a specified criteria
- There are two types of Dependabot auto-triage rules:
 - GitHub-curated default rules
 - Custom auto-triage rules
- Auto-dismissed rules can be reopened if the alert metadata changes



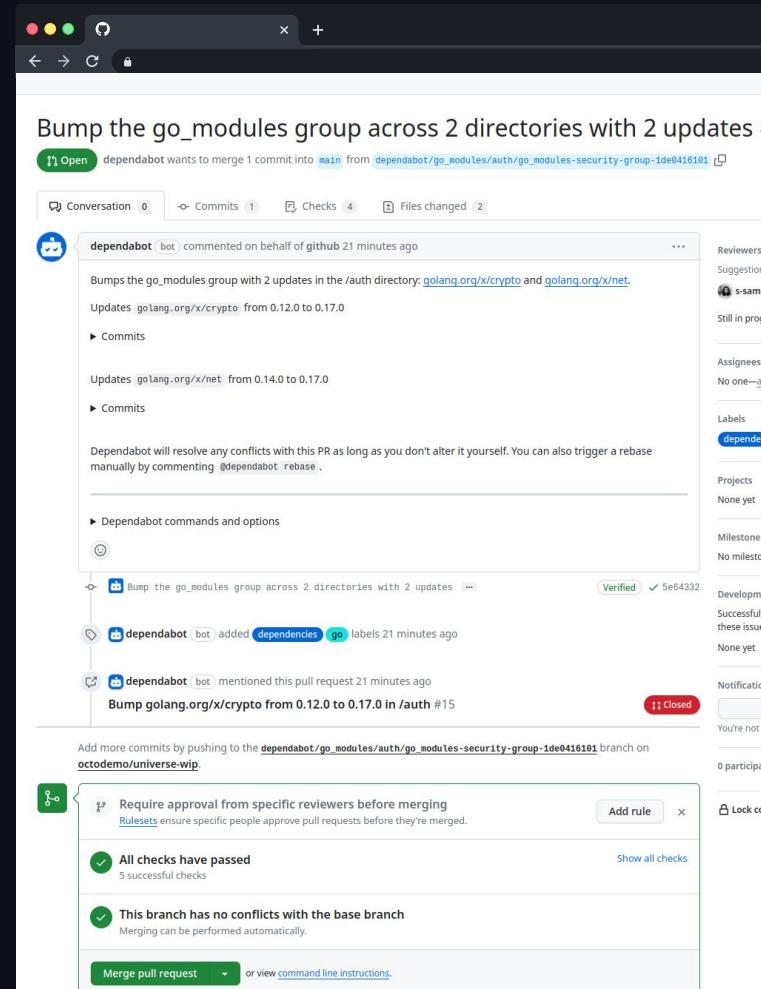
Dependabot security updates

- Dependabot raises a pull request to update the dependency to the minimum version that includes the patch and links the pull request to the Dependabot alert, or reports an error on the alert
- Each pull request includes information about the vulnerability like release notes, changelog entries, and commit details. Details of which vulnerability a pull request resolves are hidden from anyone who does not have access to Dependabot alerts for the repository
- Pull request raised to default branch. This cannot be configured



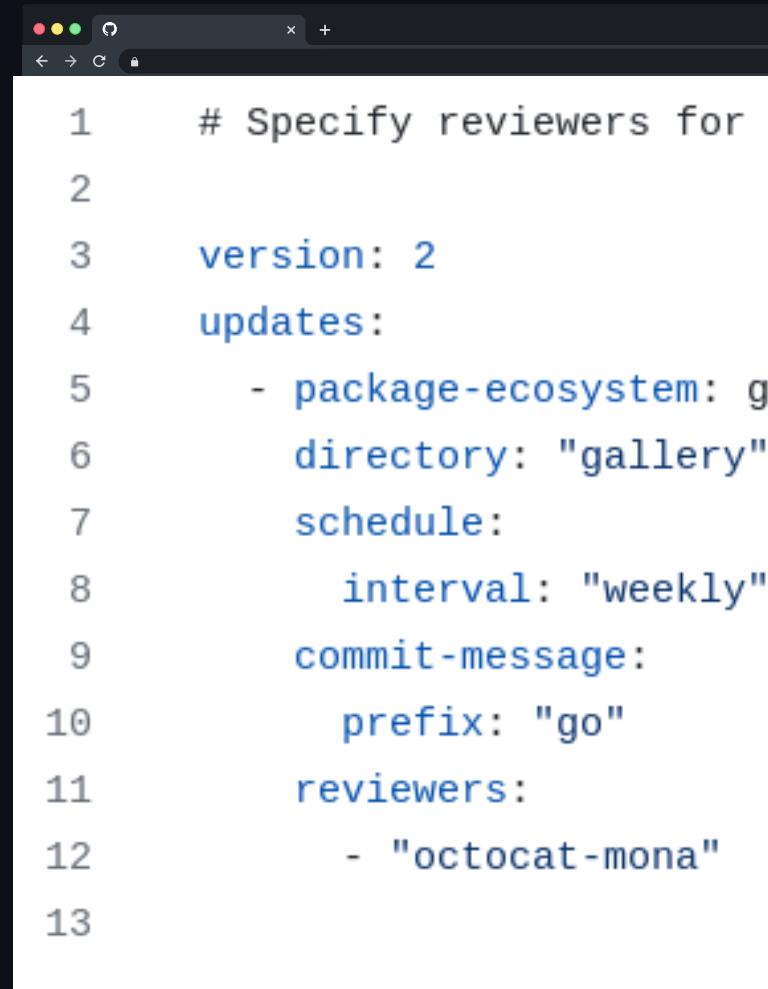
Grouped security updates

- To reduce the number of pull requests, grouped security updates can be enabled which groups sets of dependencies together (per package ecosystem) and raises a PR.
- Dependabot will group dependencies from different directories. Dependabot will not group dependencies from different package ecosystems together, and it will not group security updates with version updates



Dependabot version updates

- Keeps your packages updated to the latest version by raising a pull request to update the manifest to the latest version of the dependency.
- Enabled through a `dependabot.yml` configuration file
- Looks at the semantic versioning (semver) of the dependency to decide whether it should update to that version



```
1 # Specify reviewers for
2
3 version: 2
4 updates:
5   - package-ecosystem: github
6     directory: "gallery"
7     schedule:
8       interval: "weekly"
9     commit-message:
10    prefix: "go"
11  reviewers:
12    - "octocat-mona"
13
```

Configuring Dependabot

- Dependabot can be configured via the Dependabot.yml file
- Must be stored in the .github directory of the repository's default branch
- Configure it to access private registries

```
1  # Specify reviewers for
2
3  version: 2
4  updates:
5    - package-ecosystem: github
6      directory: "gallery"
7      schedule:
8        interval: "weekly"
9      commit-message:
10        prefix: "go"
11      reviewers:
12        - "octocat-mona"
13
```

Configuring Dependabot

- These options fit broadly into the following categories:
 - Essential set up options
 - Options to customize the update schedule
 - Options to control which dependencies are updated
 - Options to add metadata to pull requests
 - Options to change the behavior of the pull requests

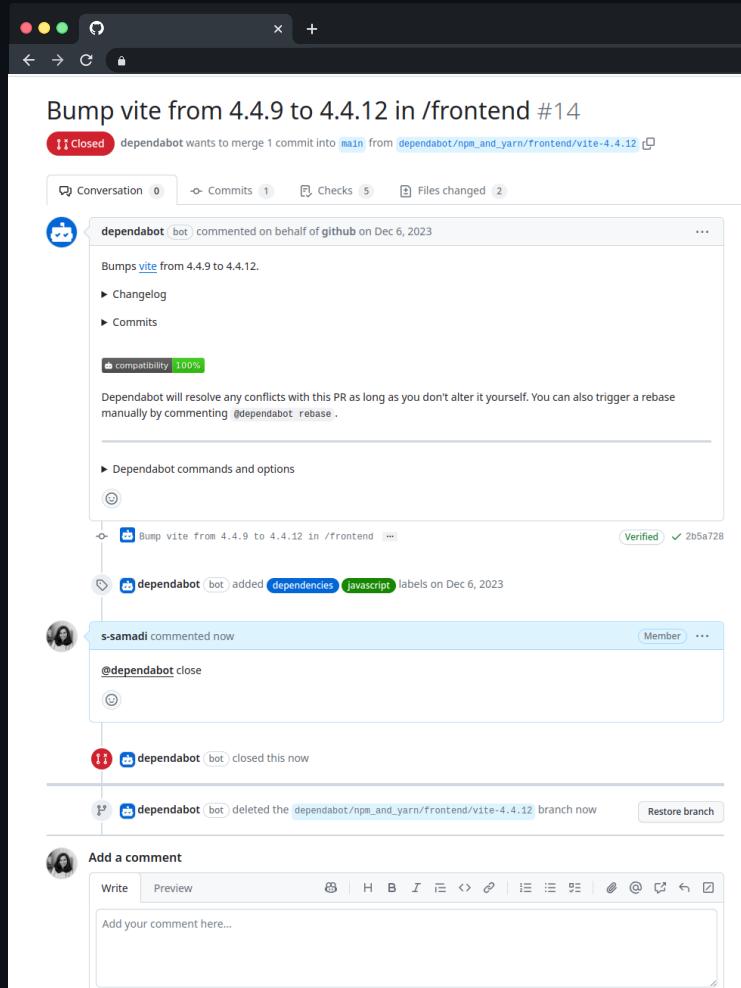


A screenshot of a macOS terminal window titled "Terminal". The window shows a configuration file for Dependabot. The code is written in YAML and includes comments (#) and several sections (version, updates, package-ecosystem, directory, schedule, interval, commit-message, prefix, reviewers). The code is as follows:

```
1 # Specify reviewers for
2
3 version: 2
4 updates:
5   - package-ecosystem: g
6     directory: "gallery"
7     schedule:
8       interval: "weekly"
9     commit-message:
10    prefix: "go"
11  reviewers:
12    - "octocat-mona"
```

Managing Dependabot PRs

- Both security updates and version updates can be managed with comment commands in the PR
 - For example - @dependabot merge
- Comment commands are supported for grouped version updates but not grouped security updates
- Each dependabot run is captured in logs via the dependabot jobs list which is accessible via the dependency graph



Grouped version updates

- Dependabot version updates can be grouped via the dependabot.yml file
- Use the groups property. Options available:
 - dependency-type - either development or production
 - patterns - strings to match dependency name
 - exclude-patterns - if this is used, Dependabot will continue to use single PRs for version updates
 - update-types - specify semantic versioning; can be minor, patch, and major



```
# `dependabot.yml` file with customized Bundler config
# In this example, the name of the group is `dev-dependencies`
# only the `patterns` and `exclude-patterns` options are defined
version: 2
updates:
  # Keep bundler dependencies up to date
  - package-ecosystem: "bundler"
    directory: "/"
    schedule:
      interval: "weekly"
  # Create a group of dependencies to be updated together
  groups:
    # Specify a name for the group, which will be used
    # and branch names
    dev-dependencies:
      # Define patterns to include dependencies in this group
      # dependency name)
      patterns:
        - "rubocop" # A single dependency name
        - "rspec*" # A wildcard string that matches multiple names
        - "*" # A wildcard that matches all dependencies in the
              # ecosystem. Note: using "*" matches everything
      # Define patterns to exclude dependencies from this group
      # dependency name)
      exclude-patterns:
        - "gc_ruboconfig"
        - "gocardless-*"
```

Dependabot - Pitfalls and Wins



You can configure Dependabot to only access private registries for a subset of ecosystems. You can find [detailed information here](#)



If your private registry is configured with an IP allow list, you can find the IP addresses Dependabot uses to access the registry in the [meta API endpoint](#)



By default, Dependabot can access public registries and additionally (if configured) can access private registries. If you want to restrict Dependabot from accessing public registries then you have to set the flag ***replaces-base*** as **true** ([doc ref](#))



When implementing branch protection rules, such as those for naming conventions, you have the option to exclude the Dependabot app using the bypass functionality. Failure to do so may result in Dependabot issues arising from branch protection rules



Dependabot security updates raises a maximum of 10 PRs to the default branch. This cannot be configured.



By default, Dependabot automatically rebases pull requests to resolve any conflicts. If a pull request has not been merged for 30 days, Dependabot will stop rebasing the pull request.



Dependabot will prioritize newer alerts when generating security update PRs



The Dependabot job logs list is accessible from the dependency graph of a repository. From the dependency graph, click the Dependabot tab, then to the right of the affected manifest file, click Recent update jobs.



If you only require security updates and want to exclude version updates, you can set `open-pull-requests-limit` to 0 in order to prevent version updates for a given package-ecosystem.



Q & A

Publishing 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 page header says "Thanks for helping make GitHub safe for everyone." Below it is a section titled "Security" which states: "GitHub takes the security of our software products and services seriously, including all of the open source repositories we host. If you find a security vulnerability in one of our repositories, please report it through our GitHub organizations, such as [GitHub](#)." A note below says: "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." A section titled "Reporting Security Issues" provides instructions: "If you believe you have found a security vulnerability in any GitHub-owned repository, please report it to us via email at opensource-security@github.com, s-samadi@github.com, or security@github.com. Instead, please send an email to opensource-security@github.com, s-samadi@github.com, or security@github.com. Please include as much of the information listed below as you can to help us better understand and address the issue." A bulleted list follows: "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)", and "Impact of the issue, including how an attacker might exploit the issue". A note at the bottom says: "This information will help us triage your report more quickly." A final section titled "Policy" links to "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 has a light gray background with a white main content area. At the top, there's a header with the title and a note: "After the draft security advisory is open, you can privately discuss it with collaborators and create a temporary private fork where you can fix the issue." Below this is a section titled "Advisory Details" with fields for "Title" (a required field marked with an asterisk), "CVE identifier" (set to "Request CVE ID later"), and "Description" (a rich text editor with "Write" and "Preview" tabs, containing sections for Impact, Patches, Workarounds, and References). At the bottom, there's a section for "Affected products" with fields for "Ecosystem" (with a dropdown menu "Select an ecosystem" and a placeholder "e.g. example.js") and "Affected versions" (with a placeholder "e.g. < 1.2.3"). There are also sections for "Patched versions" (placeholder "e.g. 1.2.3") and "Severity". A blue button at the bottom right says "+ Add another affected product". The browser interface includes standard navigation buttons (back, forward, search, etc.) and a tab bar.

Security Advisories

General process:

- Create a draft security advisory, and use the draft to privately discuss the impact of the vulnerability on your project
- Privately collaborate to fix the vulnerability in a temporary private fork.
- Publish the security advisory to alert your community of the vulnerability once a patch is released.

The screenshot shows a web browser window titled "Open a draft security advisory". The page contains fields for "Advisory Details" (Title, CVE identifier, Description), "Affected products" (Ecosystem, Package name, Affected versions, Patched versions), and a "Severity" dropdown. The "Description" field is expanded to show sections for Impact, Patches, Workarounds, and References, each with a corresponding help text.

Open a draft security advisory

After the draft security advisory is open, you can privately discuss it with collaborators and create a temporary private fork where just fill out the draft security advisory and then publish it.

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

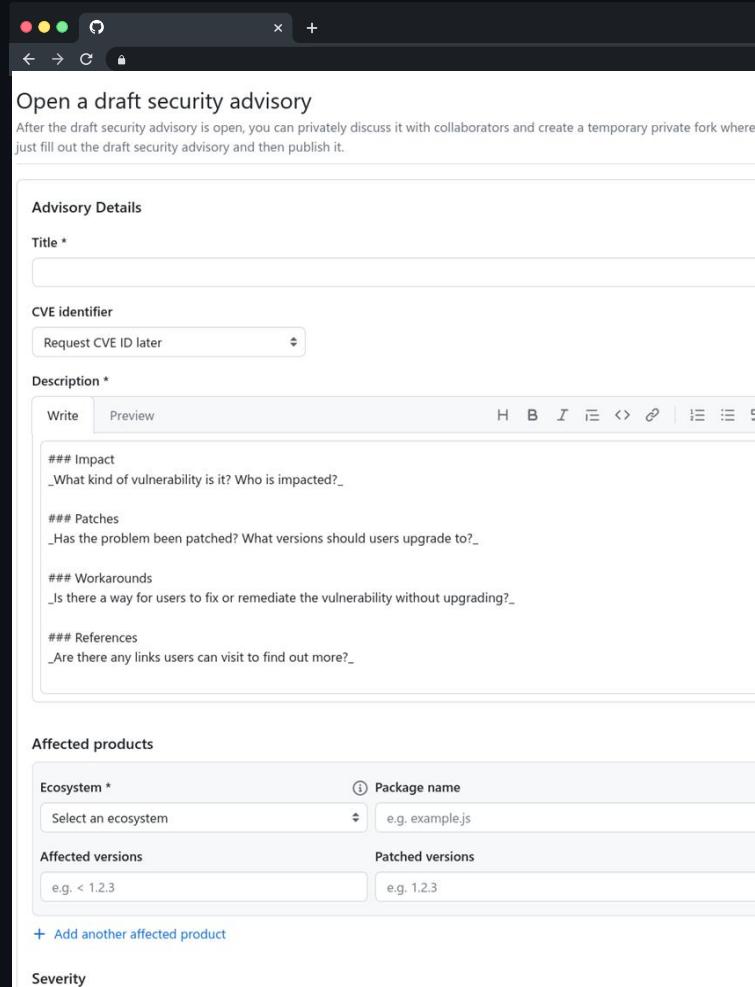
e.g. < 1.2.3 e.g. 1.2.3

+ Add another affected product

Severity

Security Advisories

- You can also use the REST API to create, list, and update repository security advisories
- You can give credit to individuals who contributed to a security advisory.
- For public repositories, once you've published the security advisory and GitHub has assigned a CVE identification number to the vulnerability, GitHub publishes the CVE to the MITRE database.





Q & A

Agenda

Introduction

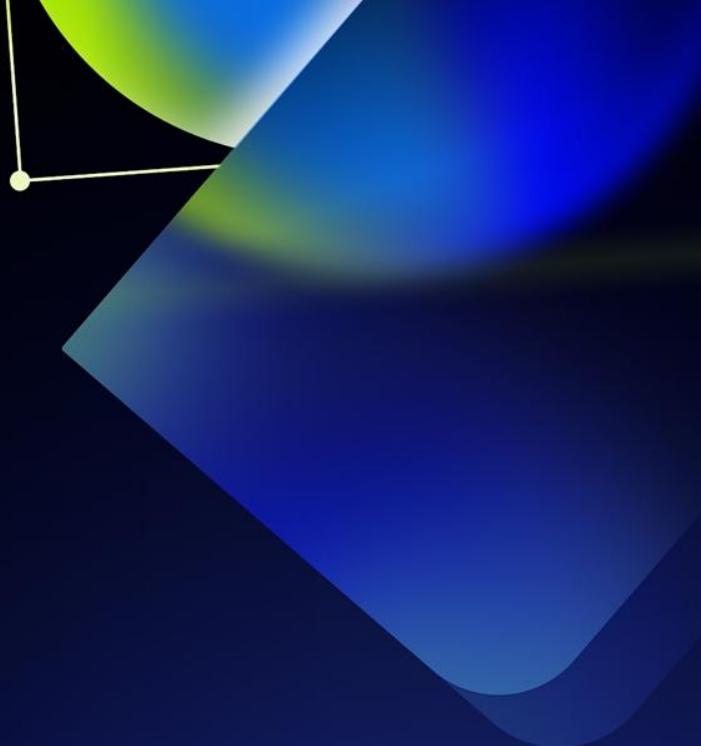
Supply Chain Security

Secret Scanning

Code Scanning

Security Overview

Secret Scanning



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

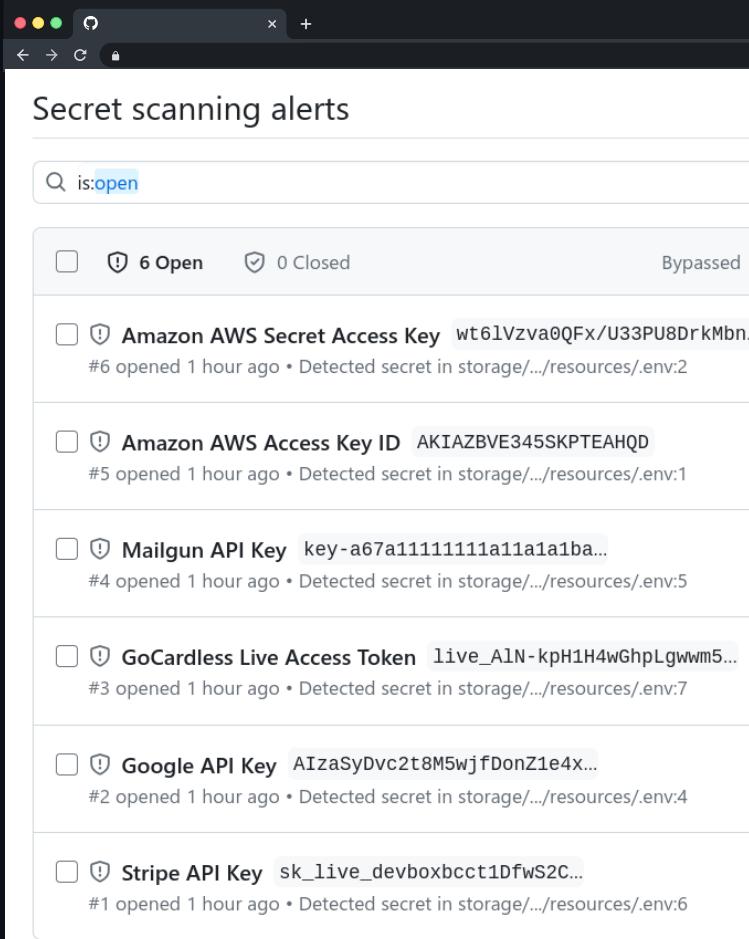
- Partner pattern and non provider pattern detection by default
- Push Protections for users and repositories
- Custom Patterns
- Regex generation with Copilot
- Generic Secrets AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD - SARIF output
- Repo rulesets
- Autofix AI
- Third-party code scanning tool support
- Security Campaigns

Secret Scanning

- Scans your entire Git history on all branches present in your GitHub repository for secrets, even if the repository is archived.
- Furthermore, it also scans descriptions, comments in issues, pull requests, and GitHub Discussions.
- Once enabled, a scan is triggered on every push. GitHub will periodically run full git history scans



The screenshot shows a web browser window with the title "Secret scanning alerts". A search bar contains the query "is:open". Below the search bar, there are two status indicators: "6 Open" and "0 Closed", with a note "Bypassed" to the right. The main content area lists six open secret scanning alerts, each with a checkbox, a shield icon, and the secret type and value. Each alert includes a timestamp indicating it was opened 1 hour ago and a note about detecting a secret in storage.

Secret Type	Value	Last Seen	Note
Amazon AWS Secret Access Key	wt61Vzva0QFx/U33PU8DrkMbn...	#6 opened 1 hour ago	Detected secret in storage/.../resources/.env:2
Amazon AWS Access Key ID	AKIAZBVE345SKPTEAHQD	#5 opened 1 hour ago	Detected secret in storage/.../resources/.env:1
Mailgun API Key	key-a67a1111111a11a1a1ba...	#4 opened 1 hour ago	Detected secret in storage/.../resources/.env:5
GoCardless Live Access Token	live_A1N-kpH1H4wGhpLgwwm5...	#3 opened 1 hour ago	Detected secret in storage/.../resources/.env:7
Google API Key	AIzaSyDvc2t8M5wjfDonZ1e4x...	#2 opened 1 hour ago	Detected secret in storage/.../resources/.env:4
Stripe API Key	sk_live_devboxbcct1DfwS2C...	#1 opened 1 hour ago	Detected secret in storage/.../resources/.env:6

Secret Scanning

- Three types of secret scanning
 - Secret Scanning for token providers
 - Secret Scanning for non-provider patterns
 - User defined custom patterns
- Can perform validity checks
- Admin privileges are required to enable secret scanning on repository, organisation, or enterprise level

The screenshot shows a web browser window titled "Secret scanning alerts". A search bar at the top contains the query "is:open". Below the search bar, there are two status filters: "6 Open" (with a shield icon) and "0 Closed" (with a checkmark icon). To the right of these filters is the word "Bypassed". The main content area displays a list of six open secret scanning alerts, each with a checkbox, a shield icon, and a secret name followed by its value and a timestamp. The alerts are:

- Amazon AWS Secret Access Key wt61Vzva0QFx/U33PU8DrkMbn... #6 opened 1 hour ago • Detected secret in storage/.../resources/.env:2
- Amazon AWS Access Key ID AKIAZBVE345SKPTEAHQD #5 opened 1 hour ago • Detected secret in storage/.../resources/.env:1
- Mailgun API Key key-a67a1111111a11a1a1ba... #4 opened 1 hour ago • Detected secret in storage/.../resources/.env:5
- GoCardless Live Access Token live_A1N-kpH1H4wGhpLgwmm5... #3 opened 1 hour ago • Detected secret in storage/.../resources/.env:7
- Google API Key AIzaSyDvc2t8M5wjfDonZ1e4x... #2 opened 1 hour ago • Detected secret in storage/.../resources/.env:4
- Stripe API Key sk_live_devboxbcct1DfwS2C... #1 opened 1 hour ago • Detected secret in storage/.../resources/.env:6

Custom Patterns

- Use regular expressions to define custom patterns detected as a part of secret scanning
- Repository, Organisation and Enterprise level
- Secret Scanning uses the Hyperscan library which only supports the Hyperscan regex construct. This is a subset of the PCRE syntax.
- Custom pattern specifiers:
 - Secret format
 - Before secret
 - After secret
 - Additional match requirements

The screenshot shows a web browser window titled "Security & analysis / New custom pattern". The interface includes fields for "Pattern name" (containing "A distinctive name for the pattern"), "Secret format (specified as a regular expression)" (containing "example_[A-Za-z0-9]{40}"), and "Test string" (with a placeholder "Provide a sample test string to make sure your configuration matches the patterns you expect"). A green button at the bottom right is labeled "Save and dry run".

Pattern name *

A distinctive name for the pattern

This cannot be edited after saving.

Secret format (specified as a regular expression) *

example_[A-Za-z0-9]{40}

The pattern for the secret, specified as a regular expression. [Learn more about defining custom patterns](#)

More options

Test string *

Provide a sample test string to make sure your configuration matches the patterns you expect

Save and dry run

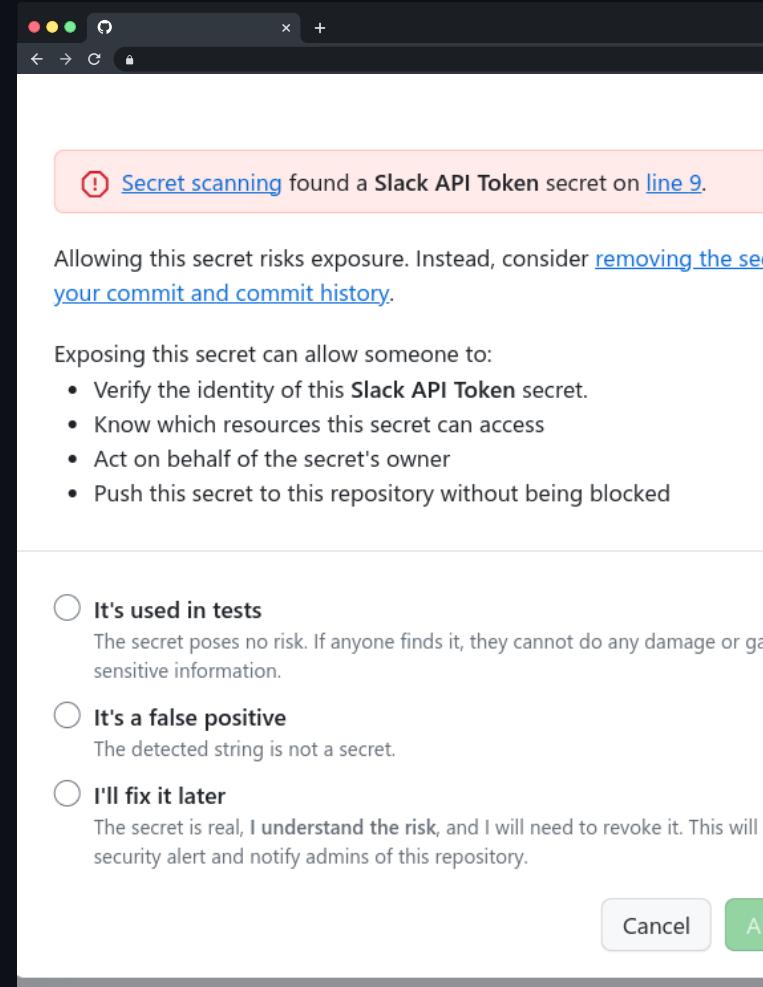
Custom Patterns + AI

- The regular expression generator allows for the generation of custom patterns without requiring knowledge of regular expressions
- Input - textual description of pattern to identify and option example string
- Output - up to 3 regular expressions

The screenshot shows a web browser window with the title "Generate regular expression" and a "Beta" status indicator. The main input field contains the text: "I want a regular expression that * finds a valid URL that starts with http or https and contains the CompanyXYZ". Below this, under "Examples of what I am looking for", there is an example URL: "http://companyXYZ.com, https://companyxyz.com/". A note at the bottom states: "This AI-powered feature may produce inaccurate results. Double-check the expressions generated and make any necessary adjustments." At the bottom right is a button labeled "Generate suggestions".

Push Protections

- Prevents secrets from being pushed into your repository for high confidence secrets
- Set at repository or organisation level
- Push protections for custom patterns can be enabled on the organisation and repository level
- Delegated bypass for push protection allows you to manage how contributors can bypass push protection rules, adding an approval process for those not on a designated bypass list.



Generic Secrets + AI

- Generic secret detection is an AI-powered expansion of secret scanning that identifies unstructured secrets (passwords) in your source code and then generates an alert
- Potential for false positives and false negatives
- Scope limited to Git repositories

The screenshot shows a web browser window titled "Secret scanning alerts". The search bar contains the query "is:open confidence:other". Below the search bar, there are filters for "Bypassed", "Validity", and "Secret type". The results section displays two open alerts:

- #9** Password secretsecret1234secretsec...
#8 opened 1 hour ago • Detected secret in storage/.../resources/application.properties:2
- #8** Password mona_value_abc124
#8 opened 1 hour ago • Detected secret in storage/.../resources/application.properties:5

Generic Secrets Limitations

Limited Scope

- AI-powered generic secret detection currently only scans for passwords in Git content.
- Does not detect other types of secrets.
- Non-git content (e.g., GitHub Issues) is excluded from the scan.

Potential for Incomplete Reporting

- May generate more false positives compared to existing secret scanning
- May miss credentials checked into repositories.

Evaluation Process

- Subject to Responsible AI Red Teaming.
- GitHub will continue monitoring the feature's efficacy and safety over time.

The screenshot shows a web interface for managing secret scanning alerts. At the top, there is a search bar with the query "is:open confidence:other". Below the search bar, there are two status filters: "2 Open" (with a shield icon) and "0 Closed" (with a checkmark icon). To the right of these filters are dropdown menus for "Bypassed", "Validity", and "Secret type". The main area displays a list of detected secrets:

- Password** secretsecret1234secretsec...
#9 opened 1 hour ago • Detected secret in storage/.../resources/application.properties:2
- Password** mona_value_abc124
#8 opened 1 hour ago • Detected secret in storage/.../resources/application.properties:5

Secret Scanning Notifications

Historical Scans

Notifications are sent out to:

- Organization owners, enterprise owners, and security managers— whenever a historical scan is complete, even if no secrets are found.
- Repository administrators, security managers, and users with custom roles with read/write access— whenever a historical scan detects a secret, and according to their notification preferences.
- We do not notify commit authors.

Incremental Scans

Notifications are sent out to:

- Repository administrators
- Security managers
- Users with custom roles with read/write access
- Organization owners and enterprise owners, if they are administrators of repositories where secrets were leaked
- Commit authors who've accidentally committed secrets will be notified, regardless of their notification preferences



Q & A

Agenda

Introduction

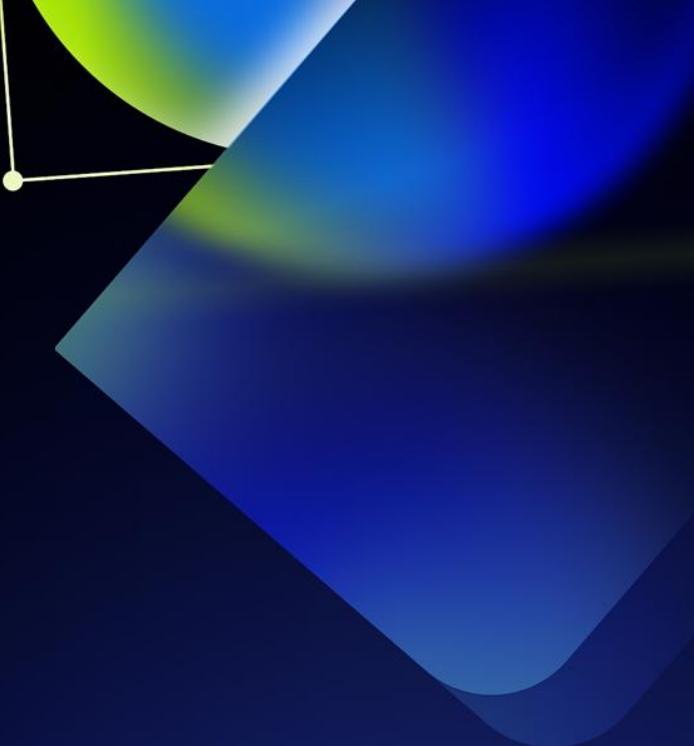
Supply Chain Security

Secret Scanning

Code Scanning

Security Overview

Code Scanning



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 Copilot
- Generic Secrets AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD - SARIF output
- Repo rulesets
- Autofix AI
- Third-party code scanning tool support
- Security Campaigns

← Security Overview Dashboard →

Code Scanning

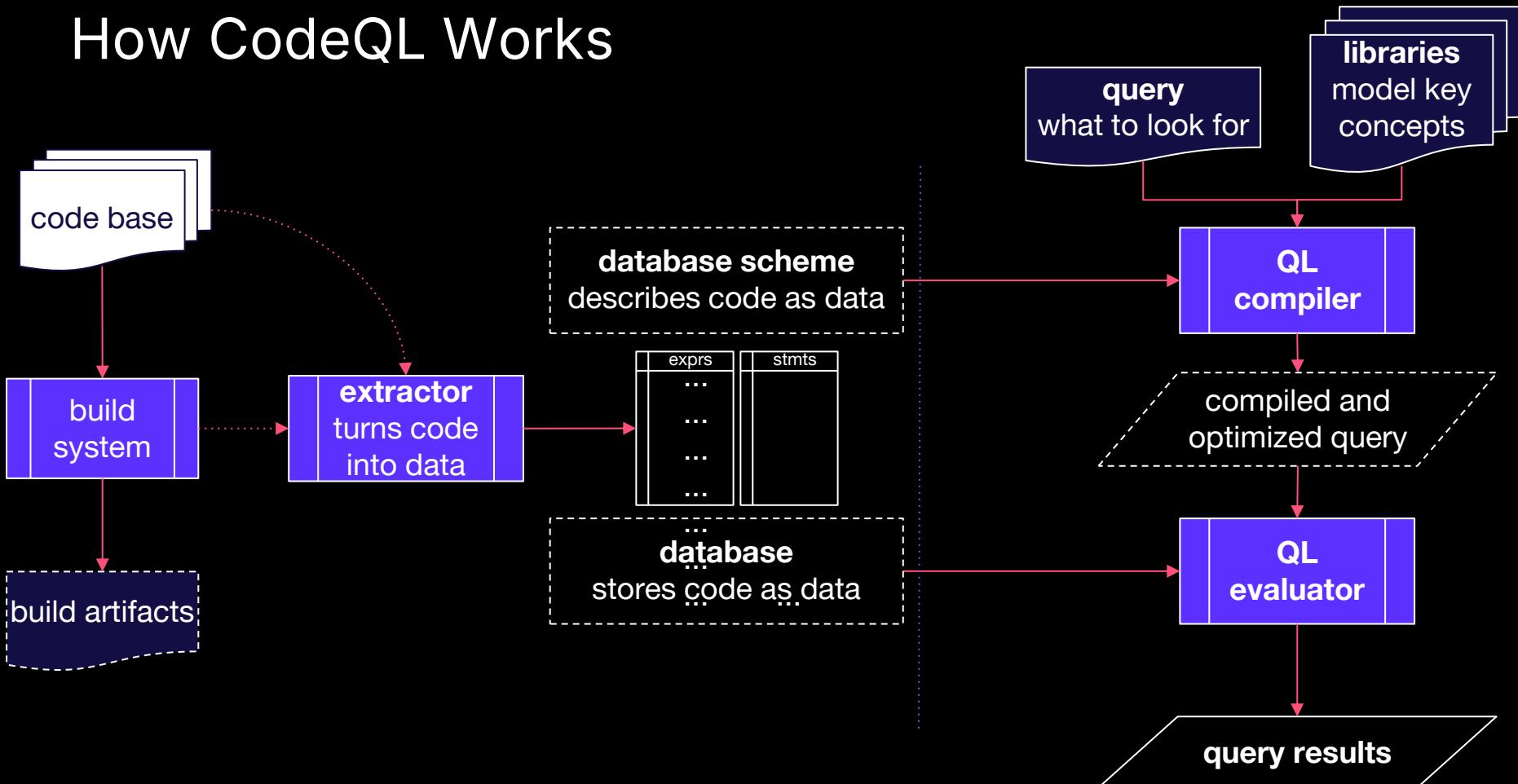
- Static analysis is a process that allows you to analyze an application's code for potential issues without executing the code itself
- GitHub code scanning allows you to automate scanning source code to prevent vulnerabilities and catch them earlier in the development pipeline
- Alerts can be viewed in the GitHub UI in a designated tab and also on the PR directly in the developer workflow
- At Github, we perform static analysis in code scanning via CodeQL, our semantic analysis engine

The screenshot shows the GitHub Code scanning interface. At the top, there is a header with the title "Code scanning" and a warning message: "⚠️ CodeQL is reporting warnings. Check the [tool status](#) for help." Below the header, there is a search bar with the query "Q: is:open branch:main". Underneath the search bar, there is a summary: "8 Open" and "0 Closed". On the right side of the interface, there is a "Language" dropdown menu. The main area displays a list of detected vulnerabilities, each with a shield icon, a warning icon, the issue type, and its severity level in a red box. The issues listed are:

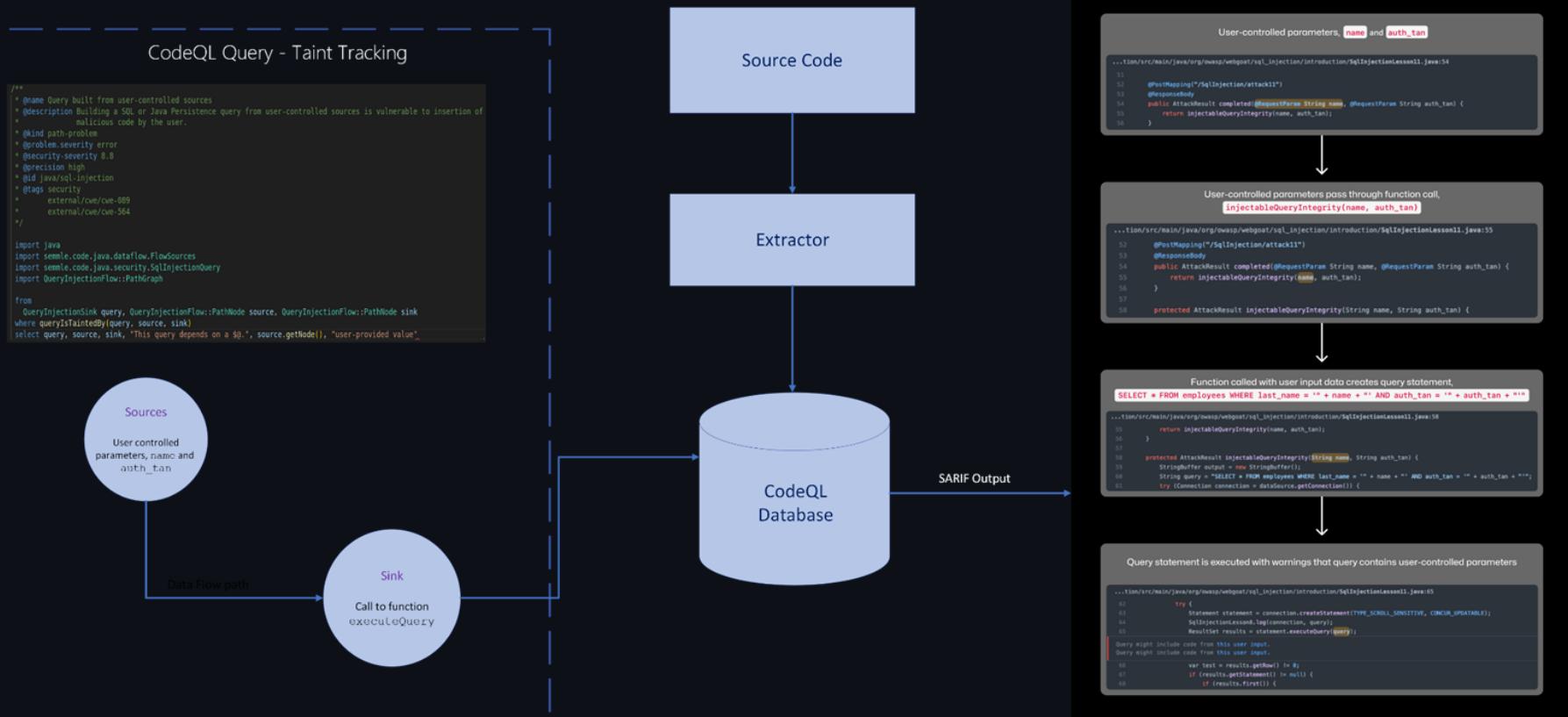
- Use of externally-controlled format string (Critical)
- Flask app is run in debug mode (High)
- Clear-text logging of sensitive information (High)
- Clear-text logging of sensitive information (High)
- Database query built from user-controlled sources (High)
- Database query built from user-controlled sources (High)
- Client-side cross-site scripting (High)
- Client-side cross-site scripting (High)

Each issue entry includes a timestamp and a note indicating it was detected by CodeQL.

How CodeQL Works



How CodeQL Works - SQL Injection Example



CodeQL

- CodeQL treats code as data. A database of the codebase is created and CodeQL queries are used to query this database
- These queries are divided into 3 suites - code scanning (default), security-extended, security-and-quality
- Three main ways to run CodeQL
 - Use default setup - automatically chooses language to analyze, query suite to run and events that trigger the scan
 - Advanced setup - GitHub Action workflow
 - Run CodeQL CLI in your external CI/CD and upload result to GitHub

The screenshot shows a web interface for Code scanning. At the top, there's a header with a warning icon and the text "CodeQL is reporting warnings. Check the [tool status](#) for help." Below the header, a search bar contains the query "is:open branch:main". Underneath, a summary shows "8 Open" issues and "0 Closed". The main content area lists eight specific findings, each with a checkbox, a shield icon, and a warning icon:

- Use of externally-controlled format string (Critical)
#3 opened 8 minutes ago • Detected by CodeQL in storage/.../controllers/BlobController.java:51
- Flask app is run in debug mode (High)
#8 opened 7 minutes ago • Detected by CodeQL in auth-ext/app.py:85
- Clear-text logging of sensitive information (High)
#7 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:678
- Clear-text logging of sensitive information (High)
#6 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:660
- Database query built from user-controlled sources (High)
#5 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:309
- Database query built from user-controlled sources (High)
#4 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:200
- Client-side cross-site scripting (High)
#2 opened 9 minutes ago • Detected by CodeQL in frontend/.../components/NotFound.vue:4
- Client-side cross-site scripting (High)
#1 opened 9 minutes ago • Detected by CodeQL in frontend/.../components/Login.vue:47

At the bottom right, there's a "ProTip!" note: "The libraries and queries that power CodeQL are open-source".

Default Setup

After enabling default setup, the code in your repository will be scanned:

- on each push to the repository's default branch, or any protected branch.
- when creating or committing to a pull request based against the repository's default branch, or any protected branch
- on a weekly schedule.

Your repository is eligible for default setup for code scanning if:

- it includes at least one CodeQL-supported language.
- GitHub Actions are enabled - supports both GitHub hosted runners and self hosted runner
- it is publicly visible, or GitHub Advanced Security is enabled.

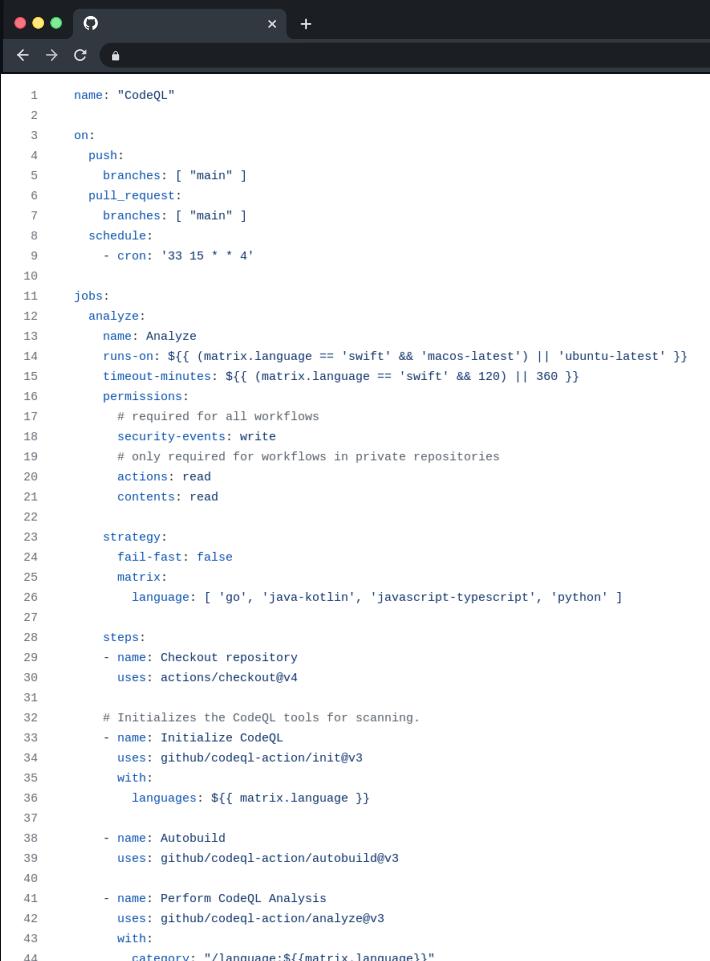
The screenshot shows the GitHub Code scanning interface. At the top, there is a header with the title "Code scanning". Below the header, a message states "CodeQL is reporting warnings. Check the [tool status](#) for help." A search bar contains the query "is:open branch:main". Below the search bar, there is a summary: "8 Open" (with a shield icon) and "0 Closed". On the right side of the interface, there is a "Language" dropdown menu. The main area displays a list of detected issues:

- Use of externally-controlled format string (Critical)
#3 opened 8 minutes ago • Detected by CodeQL in storage/.../controllers/BlobController.java:51
- Flask app is run in debug mode (High)
#8 opened 7 minutes ago • Detected by CodeQL in auth-ext/app.py:85
- Clear-text logging of sensitive information (High)
#7 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:678
- Clear-text logging of sensitive information (High)
#6 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:660
- Database query built from user-controlled sources (High)
#5 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:309
- Database query built from user-controlled sources (High)
#4 opened 8 minutes ago • Detected by CodeQL in gallery/main.go:200
- Client-side cross-site scripting (High)
#2 opened 9 minutes ago • Detected by CodeQL in frontend/.../components/NotFound.vue:4
- Client-side cross-site scripting (High)
#1 opened 9 minutes ago • Detected by CodeQL in frontend/.../components/Login.vue:47

💡 **ProTip!** The libraries and queries that power CodeQL are open-sour...

Advanced Setup

- GitHub Actions workflow - YAML file
- Main components in files
 - Trigger points
 - Push
 - Pull request
 - Schedule (default branch)
 - Workflow Dispatch
 - Path-ignore (when not what).
 - Operating System: Linux, Windows, OS
 - CodeQL Action Init - creates CodeQL database, input query suite and language
 - Input custom build steps if manual
 - CodeQL Action Analyze - run CodeQL queries on database



```
1  name: "CodeQL"
2
3  on:
4    push:
5      branches: [ "main" ]
6    pull_request:
7      branches: [ "main" ]
8    schedule:
9      - cron: '33 15 * * 4'
10
11  jobs:
12    analyze:
13      name: Analyze
14      runs-on: ${{ matrix.language == 'swift' && 'macos-latest' }|| 'ubuntu-latest'}
15      timeout-minutes: ${{ matrix.language == 'swift' && 120 }|| 360 }
16      permissions:
17        # required for all workflows
18        security-events: write
19        # only required for workflows in private repositories
20        actions: read
21        contents: read
22
23      strategy:
24        fail-fast: false
25        matrix:
26          language: [ 'go', 'java-kotlin', 'javascript-typescript', 'python' ]
27
28      steps:
29        - name: Checkout repository
30          uses: actions/checkout@v4
31
32        # Initializes the CodeQL tools for scanning.
33        - name: Initialize CodeQL
34          uses: github/codeql-action/init@v3
35          with:
36            languages: ${{ matrix.language }}
37
38        - name: Autobuild
39          uses: github/codeql-action/autobuild@v3
40
41        - name: Perform CodeQL Analysis
42          uses: github/codeql-action/analyze@v3
43          with:
44            category: "/language:${{matrix.language}}"
```

CodeQL CLI

- Standalone, command-line tool
- 3 commands to generate analysis results and upload on GitHub:
 - database create
 - database analyze
 - github upload results

```
# Create CodeQL databases for Java and Python in the 'codeql-dbs' directory
# Call the normal build script for the codebase: 'myBuildScript'

codeql database create codeql-dbs --source-root=src \
    --db-cluster --language/java,python --command=./myBuildScript

# Analyze the CodeQL database for Java, 'codeql-dbs/java'
# Tag the data as 'java' results and store in: 'java-results.sarif'

codeql database analyze codeql-dbs/java java-code-scanning.qls \
    --format=sarif-latest --sarif-category=java --output=java-results.sarif

# Analyze the CodeQL database for Python, 'codeql-dbs/python'
# Tag the data as 'python' results and store in: 'python-results.sarif'

codeql database analyze codeql-dbs/python python-code-scanning.qls \
    --format=sarif-latest --sarif-category=python --output=python-results.sarif

# Upload the SARIF file with the Java results: 'java-results.sarif'
# The GitHub App or personal access token created for authentication
# with GitHub's REST API is available in the 'GITHUB_TOKEN' environment variable.

codeql github upload-results \
    --repository=my-org/example-repo \
    --ref=refs/heads/main --commit=deb275d2d5fe9a522a0b7bd8b6b6a1c939552718 \
    --sarif=java-results.sarif

# Upload the SARIF file with the Python results: 'python-results.sarif'

codeql github upload-results \
    --repository=my-org/example-repo \
    --ref=refs/heads/main --commit=deb275d2d5fe9a522a0b7bd8b6b6a1c939552718 \
    --sarif=python-results.sarif
```

Autofix AI

- Code scanning autofix, powered by GitHub Copilot, enhances CodeQL's capabilities by providing automatic, targeted suggestions to resolve code scanning alerts. It helps prevent new security vulnerabilities by leveraging large language models (LLMs) to generate potential fixes based on codebase data and CodeQL analysis
- Converts alert descriptions and locations into actionable code changes using GPT-4o from OpenAI
- Enabled by default for repositories using CodeQL; can be disabled at the enterprise, organization, or repository level
- Autofix supports C#, C/C++, Go, Java/Kotlin, Swift, JavaScript/TypeScript, Python, and Ruby.

github-advanced-security (bot) found potential problems on Oct 31, 2023

frontend/src/components/Gallery.vue

```
235 +     return "";
236 +   }
237 +   //escape all occurrences of apostrophe
238 +   input = input.replace("'", "'");

```

✗ Check failure

Code scanning / CodeQL

Incomplete string escaping or encoding (High)

This replaces only the first occurrence of '''.
Show more details

Dismiss alert ▾

github-advanced-security (bot)

The vulnerability in the code is due to the use of the `replace` method in JavaScript, which only replaces the first occurrence of the specified value. In this case, it only replaces the first occurrence of the apostrophe. This can lead to security vulnerabilities as it allows for potential SQL injection attacks if the input string contains more than one apostrophe.

To fix this vulnerability, we need to use a regular expression with the `g` (global) flag in the `replace` ensure that all occurrences of the apostrophe character are replaced, not just the first one.

The change needs to be made in the `sanitizeInput` function in the `frontend/src/components/Gallery.vue` specifically on line 238.

No additional imports or declarations are needed at the top of the file.

An alternative approach would be to use a well-tested sanitization library, but this would require adding dependency to the project, which might not be desirable.

Suggested fix generated by CodeQL and AI (Beta)

```
237 237 //escape all occurrences of apostrophe
238 -   input = input.replace("'", "'");
238 +   input = input.replace(/'/g, "'");
239 239

```

Dismiss suggestion Edit ▾

This feature may produce inaccurate results. Double-check the suggested code change and make any necessary adjustments.

Reply...

Security Overview

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

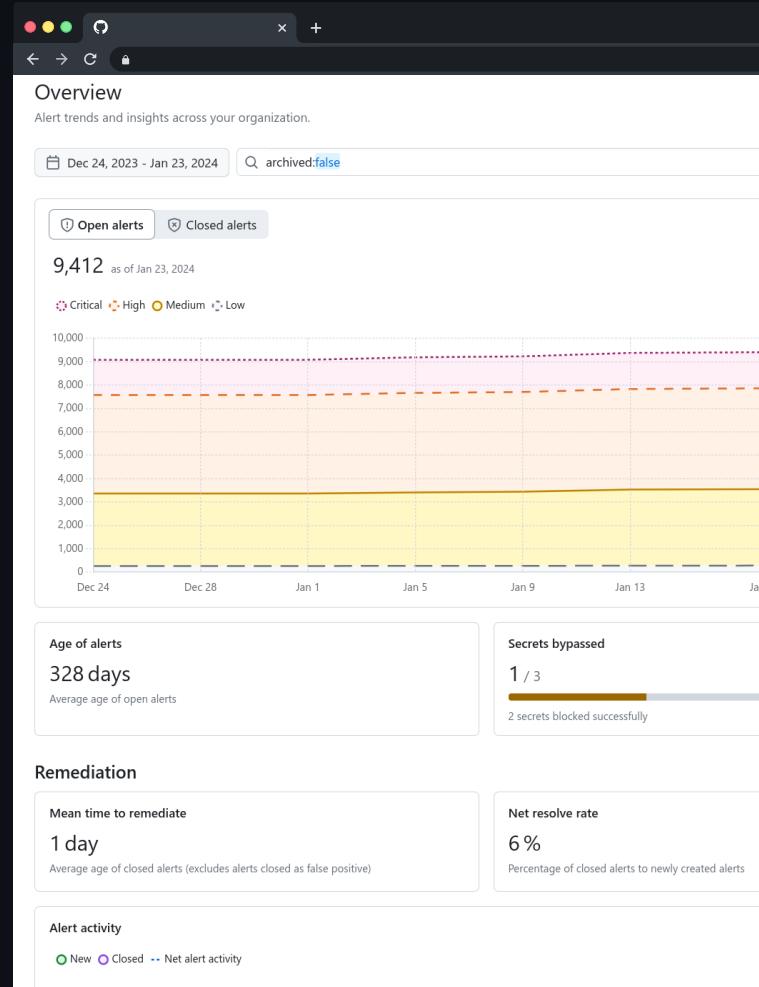
- Partner pattern and non provider pattern detection by default
- Push Protections for users and repositories
- Custom Patterns
- Regex generation with Copilot
- Generic Secrets AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD - SARIF output
- Repo rulesets
- Autofix AI
- Third-party code scanning tool support
- Security Campaigns

Security Overview

- 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





Thank you