

(1) Fix Overly Permissive File Permissions

The screenshot shows a GitHub CodeQL alert for the repository 'fraxhost / code-scanning-codeql'. The alert is titled 'Overly permissive file permissions' and is marked as 'Fixed' in the 'main' branch. The alert details include a code snippet from 'main.py:122' showing a vulnerable line: `os.chmod(path, 0o777)`. The alert is categorized as 'High' severity and is associated with the rule ID 'py/overly-permissive-file'. The alert history shows it was first detected in a commit 47 minutes ago, appeared in the 'main' branch, and was fixed in the 'main' branch. The alert is verified and associated with the weakness 'CWE-732'.

Code scanning alerts / #6

Overly permissive file permissions

Fixed in main now

```
main.py:122
119 # 16) Setting world-writable permissions on a sensitive file
120 def insecure_chmod_example(path: str) -> None:
121     # Vulnerable: setting mode to 0o777 gives write permissions to everyone
122     os.chmod(path, 0o777)
```

Overly permissive mask in chmod sets file to world writable.

CodeQL

```
123
124 # 17) Open redirect: returning user-controlled URL in a redirect
125 def open_redirect_example(next_url: str) -> str:
```

Tool	Rule ID	Query
CodeQL	py/overly-permissive-file	View source

When creating a file, POSIX systems allow permissions to be specified for owner, group and others separately. Permissions should be kept as strict as possible, preventing access to the files contents by other users.

[Show more](#)

First detected in commit 47 minutes ago

Implement examples of common Python vulnerabilities [Verified](#) ✓ 3f8879c

main.py:122 on branch main

Appeared in branch main 47 minutes ago

✓ Security — CodeQL #10: Commit 3f8879c (language: python)

Fixed in branch main now

Replace insecure chmod example with secure version [Verified](#) ✓ de16fce

Severity: High

Assignees: [Preview](#)

No one - [Assign yourself](#)

Affected branches

main default

Fixed 1 minute ago via commit de16fce

Development

No linked branches or pull requests.

Tags

security

Weaknesses

CWE-732

© 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Community](#) [Docs](#) [Contact](#) [Manage cookies](#) Do not share my personal information

Screenshot: Fix Overly Permissive File Permissions

(2) Fix Insecure Temporary File

The screenshot displays a GitHub CodeQL alert titled "Insecure temporary file" with a severity of "High". The alert is marked as "Fixed" and was detected in the `main` branch 2 minutes ago. The vulnerability is located in `main.py:103`, where the `tempfile.mktemp` function is used to generate a predictable temporary filename. The fix involves using `tempfile.mktemp(prefix="tmpvuln_")` to ensure the filename is unpredictable.

CodeQL Alert Details:

- Severity:** High
- Assignees:** No one - Assign yourself
- Affected branches:** `main` (default)
- Development:** No linked branches or pull requests.
- Tags:** security
- Weaknesses:** CWE-377

CodeQL Alert Description:

Call to deprecated function `tempfile.mktemp` may be insecure.

CodeQL

```
100 # 15) Use of tempfile.mktemp (predictable temp filename)
101 def mktemp_example() -> str:
102     # Vulnerable: mktemp is insecure (race conditions / predictable name)
103     name = tempfile.mktemp(prefix="tmpvuln_")
104
105 # Danger: attacker could create file at that path before you open it
106 with open(name, "w") as f:
107     f.write("data")
```

Tool: CodeQL
Rule ID: py/insecure-temporary-file
Query: View source

Functions that create temporary file names (such as `tempfile.mktemp` and `os.tmpnam`) are fundamentally insecure, as they do not ensure exclusive access to a file with the temporary name they return. The file name returned by these functions is guaranteed to be unique on creation but the file must be opened in a separate operation. There is no guarantee that the creation and open operations will happen atomically. This provides an opportunity for an attacker to interfere with the file before it is opened.

[Show more](#)

Alert History:

- First detected in commit 40 minutes ago
- Implement examples of common Python vulnerabilities (Verified) ✓ 3f8079c
- main.py:103 on branch `main`
- Appeared in branch `main` 40 minutes ago
- Security - CodeQL #10: Commit 3f8079c (language: python)
- Fixed in branch `main` 2 minutes ago
- Improve security of temporary file creation (Verified) ✓ f22d881

© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

Screenshot: Fix Insecure Temporary File

(3) Fix Request without Certificate Validation

The screenshot shows a GitHub Security alert for a CodeQL scan. The alert is titled "Request without certificate validation" and is marked as "Fixed" in the main branch, occurring 1 minute ago. The alert details a high-severity issue where a request is made without certificate validation because it is disabled. The code snippet shows a Python function that disables SSL certificate verification and TLS verification. The alert is associated with the rule ID "py/request-without-cert-validation" and the query "View source". The alert is verified by commit b5c6e75f and 441cf6a. The alert is also associated with the tags "security" and "CWE-295".

Code scanning alerts / #4

Request without certificate validation

Fixed in `main` 1 minute ago

Severity
High

Assignees [Preview](#)
No one - [Assign yourself](#)

Affected branches
`main` (default)
Fixed 2 minutes ago via commit 441cf6a

Development
No linked branches or pull requests.

Tags
`security`

Weaknesses
► [CWE-295](#)

```
main.py:64
61
62 # 8) Disabling SSL certificate verification
63 def disable_ssl_verification_example(url: str) -> requests.Response:
64     resp = requests.get(url, verify=False) # disabled TLS verification

This request may run without certificate validation because it is disabled.
CodeQL
65     return resp
66
67 # 9) Logging sensitive data
```

Tool CodeQL
Rule ID py/request-without-cert-validation
Query View source

Encryption is key to the security of most, if not all, online communication. Using Transport Layer Security (TLS) can ensure that communication cannot be interrupted by an interloper. For this reason, it is unwise to disable the verification that TLS provides. Functions in the `requests` module provide verification by default, and it is only when explicitly turned off using `verify=False` that no verification occurs.

[Show more](#)

First detected in commit 1 hour ago

[Add vulnerable examples for CodeQL testing](#) **Verified** ✓ b5c6e75f

`main.py:64` on branch `main`

Appeared in branch `main` 1 hour ago

✓ **Security - CodeQL #9:** Commit b5c6e75f (language: python)

Fixed in branch `main` 1 minute ago

[Update SSL verification to use secure requests](#) **Verified** ✓ 441cf6a

© 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Community](#) [Docs](#) [Contact](#) [Manage cookies](#) Do not share my personal information

Screenshot: Request without Certificate Validation

(4) Fix Use of a Broken or Weak Cryptographic Hashing Algorithm on Sensitive Data

The screenshot shows a GitHub Security alert interface. At the top, the alert title is "Use of a broken or weak cryptographic hashing algorithm on sensitive data" with a "Dismiss alert" button. Below the title, it indicates the alert is "Fixed" and was detected "1 minute ago" in the "main" branch. The main content area displays a code snippet from `main.py:39` showing a weak MD5 hashing function: `def weak_crypto_example(password: str) -> str: h = hashlib.md5() # weak hash h.update(password.encode("utf-8"))`. A CodeQL alert message states: "Sensitive data (password) is used in a hashing algorithm (MD5) that is insecure for password hashing, since it is not a computationally expensive hash function." Below this, a table provides details about the tool (CodeQL), rule ID (py/weak-sensitive-data-hashing), and query (View source). A note explains: "Using a broken or weak cryptographic hash function can leave data vulnerable, and should not be used in security related code." To the right, a sidebar shows the severity as "High", no assignees, affected branches (main, default), and development status. A "Weaknesses" section lists CWE-327, CWE-328, and CWE-916. The bottom section shows the alert's history: it was first detected in a commit 1 hour ago, added to the branch, and then fixed in the same branch 1 minute ago by implementing secure password hashing with bcrypt.

Code scanning alerts / #3

Use of a broken or weak cryptographic hashing algorithm on sensitive data

Dismiss alert

Fixed in main 1 minute ago

```
main.py:39
36 # 3) Weak cryptography usage (MD5)
37 def weak_crypto_example(password: str) -> str:
38     h = hashlib.md5() # weak hash
39     h.update(password.encode("utf-8"))
```

Sensitive data (password) is used in a hashing algorithm (MD5) that is insecure for password hashing, since it is not a computationally expensive hash function.

CodeQL [Show paths](#)

```
40     return h.hexdigest()
41
42 # 4) Insecure random for security-sensitive token
```

Tool	Rule ID	Query
CodeQL	py/weak-sensitive-data-hashing	View source

Using a broken or weak cryptographic hash function can leave data vulnerable, and should not be used in security related code.

[Show more](#)

First detected in commit 1 hour ago

Add vulnerable examples for CodeQL testing [Verified](#) [b6cce75](#)

main.py:39 on branch main

Appeared in branch main 1 hour ago

Security — CodeQL #9: Commit b6cce75f (language: python)

Fixed in branch main 1 minute ago

Implement secure password hashing with bcrypt [Verified](#) [97fda2a](#)

Severity: High

Assignees: [Preview](#) No one - [Assign yourself](#)

Affected branches: main, default Fixed 1 minute ago via commit 97fda2a

Development: No linked branches or pull requests.

Tags: security

Weaknesses: CWE-327, CWE-328, CWE-916

© 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Community](#) [Docs](#) [Contact](#) [Manage cookies](#) Do not share my personal information

Screenshot: Use of a Broken or Weak Cryptographic Hashing Algorithm on Sensitive Data

(5) Fix Clear-Text Logging of Sensitive Information

The screenshot shows a GitHub CodeQL alert titled "Clear-text logging of sensitive information" with a severity of "High". The alert is marked as "Fixed" and was detected in the "main" branch 1 minute ago. The code snippet shows a logging function that logs sensitive data (password) as clear text. The alert is associated with the rule ID "py/clear-text-logging-sensitive-data" and the query "View source". The alert is also associated with the tags "security" and "vulnerability". The alert is also associated with the tags "vulnerability" and "security". The alert is also associated with the tags "vulnerability" and "security".

Code scanning alerts / #2

Clear-text logging of sensitive information

Fixed in main 1 minute ago

Severity: High

Assignees: Preview

No one - Assign yourself

Affected branches: main (default) Fixed 2 minutes ago via commit 3d42b2a

Development: No linked branches or pull requests.

Tags: security

Weaknesses: CWE-312, CWE-359, CWE-532

Tool: CodeQL Rule ID: py/clear-text-logging-sensitive-data Query: View source

If sensitive data is written to a log entry it could be exposed to an attacker who gains access to the logs.

Show more

First detected in commit 1 hour ago

Add vulnerable examples for CodeQL testing Verified b6c6e75f

main.py:69 on branch main

Appeared in branch main 1 hour ago

Security - CodeQL #9: Commit b6c6e75f (language: python)

Fixed in branch main 1 minute ago

Rename and secure sensitive logging function Verified 3d42b2a

© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

Screenshot: Clear-Text Logging of Sensitive Information