

(1) Overly Permissive File Permissions

- **Overly permissive file permissions** occur when a file or directory is created with access rights that are too broad (e.g., anyone can read/write/execute it).
- This allows **unauthorized users or processes** to modify, delete, or steal sensitive data.
- In Linux, using permissions like `chmod 777` or Python's `os.chmod(path, 0o777)` is a common example.
- Attackers can exploit this to replace config files, inject malicious code, or escalate privileges.
- Secure practice: grant the **minimum necessary access** (e.g., `600` or `700`) and avoid world-writable directories unless explicitly required.

Filename: [main.py](#)
Line No: 112

The screenshot shows a GitHub CodeQL alert titled "Overly permissive file permissions" with a severity of "High". The alert is located in the file `main.py` at line 112. The code snippet shows a function `insecure_chmod_example` that sets permissions to `0o777` for a sensitive file. The alert message states: "Overly permissive mask in chmod sets file to world writable." The alert was first detected 19 minutes ago in the commit `3f8879cc`. The alert is associated with the rule `py/overly-permissive-file` and the query `View source`. The alert is also associated with the tag `security` and the weakness `CWE-732`. The alert is currently open and has a "Generate fix" button. The alert is also associated with the branch `main` and the commit `3f8879cc`. The alert is also associated with the file `main.py` and the line number 112. The alert is also associated with the repository `fraxhost / code-scanning-codeql`. The alert is also associated with the user `fraxhost`. The alert is also associated with the repository `code-scanning-codeql`. The alert is also associated with the user `fraxhost`. The alert is also associated with the repository `code-scanning-codeql`. The alert is also associated with the user `fraxhost`.

Screenshot: Overly Permissive File Permissions

(2) Insecure Temporary File

- An **insecure temporary file** is created in a way that allows another program or attacker to access or modify it before the legitimate program uses it.
- If a temp file is created with predictable names (like `/tmp/data.txt`), an attacker can create or replace it beforehand.
- This can cause data leaks, unauthorized file modification, or even **code execution** if the program later runs data from that file.
- Temp directories like `/tmp` are shared, so files placed there need secure random names and proper permissions.
- Safe practice: use secure temp functions (e.g., `tempfile.NamedTemporaryFile()` in Python) to avoid guessing and tampering.

Filename: [main.py](#)

Line No: 103

The screenshot shows a GitHub CodeQL alert titled "Insecure temporary file" with a severity of "High". The alert is located in the file `main.py` at line 103. The code snippet shows a function `mktemp_example()` that returns a predictable temporary file name using `tempfile.mktemp()`. A comment indicates this is a vulnerability because the file name is predictable. The alert also mentions that the function `tempfile.mktemp` is deprecated and may be insecure. The alert was first detected 22 minutes ago. The right sidebar shows the alert's details, including the severity, assignees, affected branches, and development status. The bottom of the alert shows the commit history and the alert's status as "Verified".

Code scanning alerts / #5

Insecure temporary file

[Open](#) in `main` 22 minutes ago

Rule `py/insecure-temporary-file` is not supported by Copilot Autofix for CodeQL [Generate fix](#)

main.py:103

```
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="tmpin_")
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")
```

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

CodeQL

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](#)

First detected in commit 22 minutes ago

[Implement examples of common Python vulnerabilities](#) [Verified](#) [3f8079c](#)

`main.py:103` on branch `main`

Appeared in branch `main` 22 minutes ago

✓ Security — CodeQL #10: Commit [3f8079c](#) (language: python)

Severity: High

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

Affected branches: `main` (default) First detected 22 minutes ago

Development: Link a branch, pull request, or create a new branch to start working on this alert.

Tags: `security`

Weaknesses: [CWE-377](#)

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

Screenshot: Insecure Temporary File

(3) Request without Certificate Validation

- **Request without certificate validation** means a program makes an HTTPS request but does **not verify the server's SSL certificate**.
- This skips the security check that ensures the server is **authentic** and not an attacker pretending to be the server.
- Without validation, the connection can be intercepted using a **man-in-the-middle attack**.
- Attackers can steal sensitive data (passwords, API keys) or inject malicious responses.
- It is often seen in code like `verify=False` in Python `requests`, and should be avoided except in controlled testing environments.

Filename: [main.py](#)
Line No: 64

The screenshot shows a GitHub CodeQL alert interface. At the top, the alert title is "Request without certificate validation" with a severity of "High". Below the title, there's a code snippet from `main.py:64` showing a function `disable_ssl_verification_example` that calls `requests.get` with `verify=False`. A message states: "This request may run without certificate validation because it is disabled." To the right of the code, there's a table with details about the rule: `py/request-without-cert-validation`. Below the code, there's a section for "First detected in commit 40 minutes ago" and a link to "Add vulnerable examples for CodeQL testing". The bottom right corner shows the alert was "Verified" with a commit hash `b6cce75f`. The interface also includes a "Dismiss alert" button, a "Generate fix" button, and a "Speed up the remediation of this alert with Copilot Autofix for CodeQL" link.

Screenshot: Request without Certificate Validation

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

- **Weak cryptographic hashes** like MD5 or SHA-1 are no longer secure for protecting sensitive data such as passwords or tokens.
- They are vulnerable to **collision attacks**, where two different inputs produce the same hash, compromising data integrity.
- Using them on passwords allows attackers to **precompute hash tables (rainbow tables)** to reverse them easily.
- Sensitive data hashed with weak algorithms can be **easily cracked or tampered**, leading to breaches or impersonation.
- Secure practice: use modern, strong hashing algorithms like **SHA-256** with salts or **password-specific hashing functions** like **bcrypt**, **scrypt**, or **Argon2**.

Filename: [main.py](#)

Line No: 39

The screenshot shows a GitHub Security alert titled "Use of a broken or weak cryptographic hashing algorithm on sensitive data". The alert is categorized as "High" severity and is located in the file `main.py` at line 39. The code snippet shows a function `weak_crypto_example` that uses `hashlib.md5()` to hash a password. The 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." The alert was first detected 41 minutes ago. The right sidebar shows the alert's details, including the affected branches (main, default) and the development branch. The bottom of the alert shows the commit `b6cce75f` and the language `python`.

Code scanning alerts / #3

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

[Open](#) in `main` 14 minutes ago

Speed up the remediation of this alert with [Copilot Autofix for CodeQL](#) [Generate fix](#)

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

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](#)

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 41 minutes ago

[Add vulnerable examples for CodeQL testing](#) [Verified](#) [b6cce75f](#)

`main.py:39` on branch `main`

Appeared in branch `main` 41 minutes ago

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

Severity: **High**

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

Affected branches
[main](#) [default](#)
First detected 41 minutes ago

Development
Link a branch, pull request, or create a new branch to start working on this alert.

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

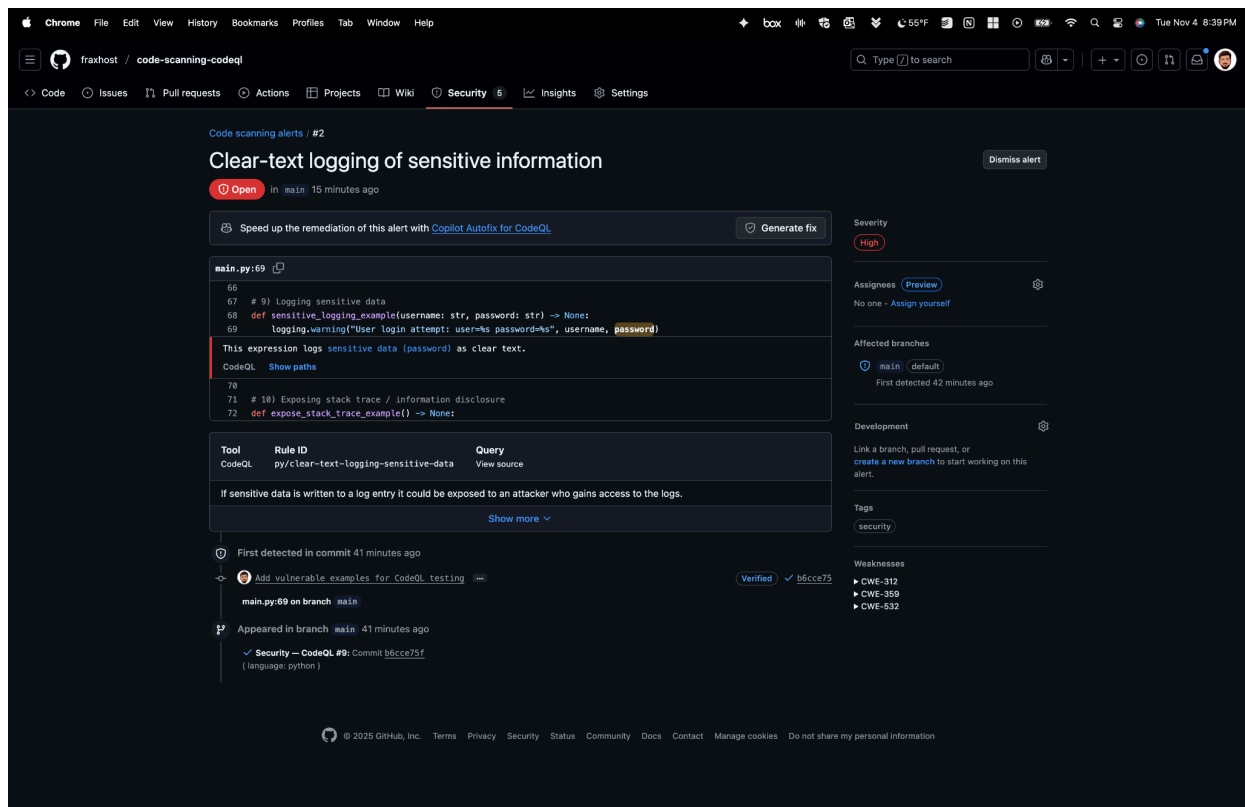
Screenshot: Use of a Broken or Weak Cryptographic Hashing Algorithm on Sensitive Data

(5) Clear-Text Logging of Sensitive Information

- **Clear-text logging** occurs when sensitive information like passwords, API keys, or personal data is written directly to logs.
- Anyone with access to the logs (developers, administrators, or attackers) can **read confidential data** easily.
- Logs may be stored long-term or sent to external systems, increasing the risk of **data leaks**.
- Logging sensitive data can lead to **compliance violations** with standards like GDPR, HIPAA, or PCI-DSS.
- Secure practice: **mask, redact, or avoid logging** sensitive information, and log only non-sensitive identifiers or anonymized data.

Filename: [main.py](#)

Line No: 69



Screenshot: Clear-Text Logging of Sensitive Information