Sensible Daten aus Git entfernen – warum einfaches Löschen nicht reicht □□

Du hast versehentlich ein Passwort, einen API-Key oder andere sensible Daten committed und vielleicht sogar auf GitHub gepusht. Das ist ein ernstes Problem, aber es lässt sich beheben – allerdings **nicht** durch einfaches Löschen der Datei in einem neuen Commit.

Warum reicht es nicht, die Datei einfach zu löschen?

Hier liegt ein fundamentales Missverständnis über Git vor, das vielen Anfängern passiert: **Git vergisst nichts**. Wenn du eine Datei mit einem Passwort commitest und sie dann im nächsten Commit löschst, ist das Passwort nicht weg – es steckt immer noch im *alten* Commit.

Jeder, der Zugriff auf dein Repository hat (oder hatte), kann:

- Die **gesamte Commit-Historie** durchsuchen
- Jeden **alten Commit auschecken** und die Datei dort finden
- Mit Befehlen wie `git log -p` oder `git show` den **kompletten Inhalt** jeder jemals existierenden Dateiversion sehen

> □□ **Merke:** Ein Commit ist wie ein Foto deines gesamten Projekts zu einem bestimmten Zeitpunkt. Auch wenn du die Datei später löschst, existiert das alte „Foto" mit der Datei noch immer in der Historie.

Das bedeutet: **Sensible Daten sind kompromittiert, sobald sie gepusht wurden** – selbst wenn du sie Sekunden später „löschst".

Was du *wirklich* tun musst

Die Lösung erfordert mehrere Schritte, und die Reihenfolge ist wichtig:

1. Sofort das kompromittierte Geheimnis ungültig machen □□

Das ist der **allerwichtigste Schritt** und sollte *sofort* passieren:

- **Passwort ändern** – logge dich ein und ändere das Passwort
- **API-Key rotieren** – erstelle einen neuen Key und deaktiviere den alten
- **Token widerrufen** – bei OAuth-Tokens, Personal Access Tokens etc.

> ⚠ **Geh davon aus, dass das Geheimnis bereits kompromittiert ist!** Bots scannen GitHub kontinuierlich nach Zugangsdaten. Innerhalb von Minuten nach dem Push können deine Daten bereits missbraucht worden sein.

2. Die Git-Historie bereinigen

Nachdem du das Geheimnis ungültig gemacht hast, solltest du es auch aus der Git-Historie entfernen – nicht weil es dadurch wieder sicher wird (das Geheimnis ist bereits kompromittiert), sondern um zu verhindern, dass es bei zukünftigen Clones oder Forks weiter verbreitet wird.

Methode A: Mit `git filter-repo` (empfohlen)

Das Tool `git filter-repo` ist der moderne, sichere Weg, um Dateien oder Inhalte aus der gesamten Git-Historie zu entfernen:

1. **Tool installieren**

```
# Mit pip (Python)
pip install git-filter-repo

# Oder über Paketmanager (z.B. Homebrew auf macOS)
brew install git-filter-repo
```

2. **Datei aus der gesamten Historie entfernen**

```
# Entfernt die Datei "config/secrets.php" aus ALLEN Commits
git filter-repo --path config/secrets.php --invert-paths
```

3. **Änderungen auf GitHub pushen**

```
# Force-Push, weil du die Historie umgeschrieben hast
git push origin --force --all
```

Methode B: Mit BFG Repo-Cleaner (einfacher für Anfänger)

Der BFG Repo-Cleaner ist speziell dafür entwickelt, sensible Daten zu entfernen:

1. **BFG herunterladen** (eine `.jar`-Datei)
2. **Passwörter aus der Historie entfernen**

```
# Erstelle eine Datei mit den zu entfernenden Texten
echo "meinGeheimesPasswort123" > passwords.txt

# BFG ausführen
java -jar bfg.jar --replace-text passwords.txt mein-repo.git
```

3. **Bereinigung abschließen und pushen**

```
cd mein-repo.git
git reflog expire --expire=now --all
git gc --prune=now --aggressive
git push origin --force --all
```

3. GitHub-Cache invalidieren

Auch nach dem Force-Push können alte Commits noch über ihre **SHA-Hashes** erreichbar sein, wenn jemand die URL kennt. GitHub bietet eine Möglichkeit, diese zu entfernen:

1. Kontaktiere den **GitHub-Support** über support.github.com
2. Bitte um die **Entfernung gecachter Ansichten** der betroffenen Commits
3. Wenn das Repository geforkt wurde, müssen auch die **Forks** bereinigt werden

4. Alle Mitwirkenden informieren

Wenn andere Personen das Repository geklont haben, müssen sie:

- Ihren lokalen Klon **löschen und neu klonen**, oder
- Mit `git fetch --all` und `git reset --hard origin/main` synchronisieren

> ⚠ Wenn jemand einen alten Klon hat und pusht, könnten die sensiblen Daten wieder ins Repository gelangen!

Zusammenfassung als Checkliste

```
flowchart TD
    A["Sensible Daten\ngepusht!"] --> B["1. Geheimnis SOFORT\nungueltigt machen"]
    B --> C["Passwort aendern\nAPI-Key rotieren\nToken widerrufen"]
    C --> D["2. Git-Historie\nbereinigen"]
    D --> E["git filter-repo\noder BFG verwenden"]
    E --> F["3. Force-Push\nauf GitHub"]
    F --> G["4. GitHub-Support\nkontaktieren"]
    G --> H["5. Team informieren\nNeu-Clone erforderlich"]

    style A fill:#ff6b6b,color:#000
    style B fill:#ffd93d,color:#000
    style C fill:#ffd93d,color:#000
```

Schritt	Aktion	Priorität
1	Geheimnis ungültig machen	◫ **Sofort**
2	Historie mit `filter-repo` oder BFG bereinigen	◫ Schnell
3	Force-Push durchführen	◫ Schnell
4	GitHub-Support kontaktieren	◫ Zeitnah
5	Team/Mitwirkende informieren	◫ Zeitnah

Wie du das in Zukunft verhinderst

Damit dir das nicht wieder passiert, solltest du einige Vorkehrungen treffen:

1. `.gitignore` **richtig konfigurieren** – sensible Dateien wie `.env`, `config/secrets.php` oder `credentials.json` sollten *niemals* getrackt werden:

   ```
   # Umgebungsvariablen und Secrets
   .env
   .env.local
   *.pem
   *.key
   config/secrets.php
   ```

2. **Umgebungsvariablen nutzen** – speichere Passwörter und API-Keys in Umgebungsvariablen statt direkt im Code:

   ```
   // ❌ Schlecht
   $password = "geheim123";

   // ✅ Gut
   $password = getenv('DB_PASSWORD');
   ```

3. **Pre-Commit-Hooks einrichten** – Tools wie git-secrets oder pre-commit können automatisch nach Secrets suchen, *bevor* du commitest
4. **GitHub Secret Scanning aktivieren** – GitHub scannt öffentliche Repositories automatisch nach bekannten Secret-Formaten und warnt dich

---

Fazit

Das versehentliche Pushen von Passwörtern ist ein häufiger und potenziell schwerwiegender Fehler. Die wichtigste Erkenntnis ist: **Einfaches Löschen reicht nicht, weil Git die komplette Historie aufbewahrt.** Deine erste Reaktion muss immer sein, das kompromittierte Geheimnis *sofort* ungültig zu machen – erst danach kümmere dich um die Bereinigung der Historie.

Revision #1
Created 2026-06-17 04:33:35 UTC by art10m
Updated 2026-06-17 04:34:37 UTC by art10m