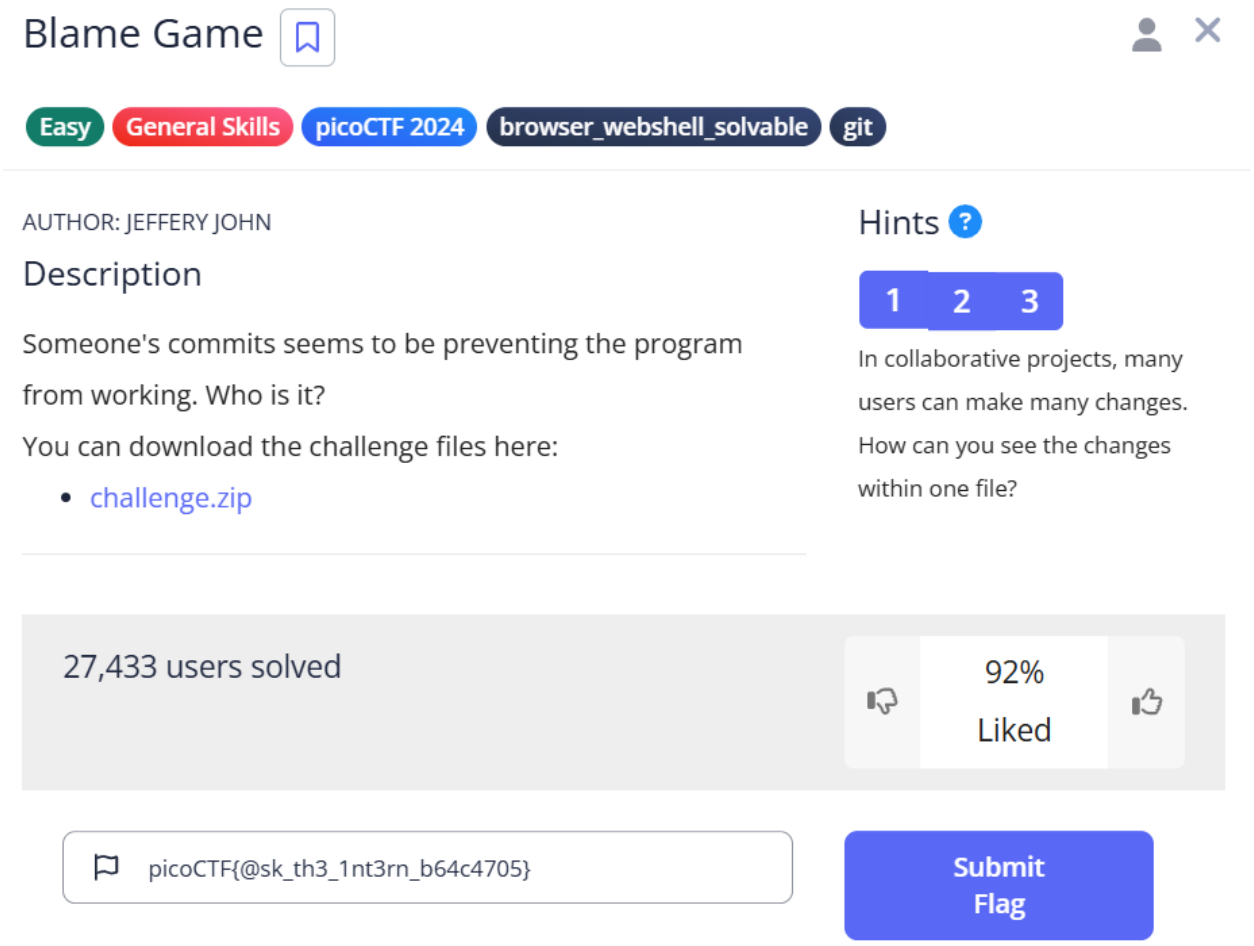
Saint Columban College
College of Computing Studies
Pagadian City

Game Blame



Author: The Analyst: Hyposelenia

Challenge: Game Blame

Category: General Skills

Date: 01/13/26

I. Objective

The objective of this task was to identify the author of a broken line in a file using Git, without running the file, and to extract the hidden flag from the commit information.

II. Background

Git is a version control system that tracks **who wrote each line** in a file and when.

Every saved change in Git is called a **commit**, and each commit stores the **author name**.

The git blame command can show the **author of every line**, which is useful when debugging or, in this case, finding a hidden flag.

We do not need to run the Python file, because the flag is hidden in the Git commit, not in the file's execution.

III. Tool Used

- **Oracle VirtualBox (Kali Linux)** – Used as the Linux environment
- **Linux Terminal** – Used to execute decoding commands
- **Git** – Used to inspect commits and authorship of the file
- C**at** – Optional, used to view the contents of the file

IV. Methodology

1. Launched Kali Linux using Oracle VirtualBox.
2. Unzip the file provided.

```
┌──(kali㉿kali)-[/media/sf_Downloaded_Cybersecfiles]
└─$ unzip challenge.zip
Archive:  challenge.zip
   creating: drop-in/
  extracting: drop-in/message.py
   creating: drop-in/.git/
   creating: drop-in/.git/branches/
  inflating: drop-in/.git/description
   creating: drop-in/.git/hooks/
  inflating: drop-in/.git/hooks/applypatch-msg.sample
  inflating: drop-in/.git/hooks/commit-msg.sample
  inflating: drop-in/.git/hooks/fsmonitor-watchman.sample
  inflating: drop-in/.git/hooks/post-update.sample
  inflating: drop-in/.git/hooks/pre-applypatch.sample
  inflating: drop-in/.git/hooks/pre-commit.sample
  inflating: drop-in/.git/hooks/pre-merge-commit.sample
  inflating: drop-in/.git/hooks/pre-push.sample
  inflating: drop-in/.git/hooks/pre-rebase.sample
  inflating: drop-in/.git/hooks/pre-receive.sample
  inflating: drop-in/.git/hooks/prepare-commit-msg.sample
  inflating: drop-in/.git/hooks/update.sample
   creating: drop-in/.git/info/
  inflating: drop-in/.git/info/exclude
   creating: drop-in/.git/refs/
   creating: drop-in/.git/refs/heads/
  extracting: drop-in/.git/refs/heads/master
   creating: drop-in/.git/refs/tags/
  extracting: drop-in/.git/HEAD
  inflating: drop-in/.git/config
   creating: drop-in/.git/objects/
   creating: drop-in/.git/objects/pack/
   creating: drop-in/.git/objects/info/
   creating: drop-in/.git/objects/7d/
  extracting: drop-in/.git/objects/7d/f869a15e76c28afb609fa4dbc059144ad70161
  extracting: drop-in/.git/objects/7d/0613c546359179460d70e0b2e9ca42136c18ee4
  extracting: drop-in/.git/objects/7d/1ca8dc38f8bc18eb35b2571d2b550673f2b415
   creating: drop-in/.git/objects/a5/
  extracting: drop-in/.git/objects/a5/6b2529881119591fce34630170f5630f4b096c
   creating: drop-in/.git/objects/f3/
  extracting: drop-in/.git/objects/f3/cec26cf7f80f91b5c3d1972f14dd4e9f97ec83
  extracting: drop-in/.git/objects/f3/cb1c88e58063947d16d222c68e04e62c23649e
  extracting: drop-in/.git/objects/f3/7406b0cab000c4af56ca105f0d194608e9402e
   creating: drop-in/.git/objects/32/
  extracting: drop-in/.git/objects/32/6544a21bf75fa38f486891c58119c236a7dbbf
  extracting: drop-in/.git/objects/32/a8c7430819aa9f2ca0a731f992c6cb8b09793c
  extracting: drop-in/.git/objects/32/ede878fe116a9ea9b2753fcf11543472e45035
  extracting: drop-in/.git/objects/32/a8b421e4870c094e436e3df403a243c9f61b19
  extracting: drop-in/.git/objects/32/223a970ba9986bd9caabcae02cb014381e5b42
```

3. Navigated to the extracted challenge folder using `cd <folder>`.

```
┌──(kali㉿kali)-[/media/sf_Downloaded_Cybersecfiles]
└─$ cd drop-in
```

4. Checked if the folder contained a Git repository using `ls -a` (looked for `.git`).

```
┌──(kali㉿kali)-[/media/sf_Downloaded_Cybersecfiles/drop-in]
└─$ ls -as
total 9
0 .   4 ..   4 .git   1 message.py
```

5. If Git complained about “dubious ownership,” ran:

   git config --global --add safe.directory /path/to/folder

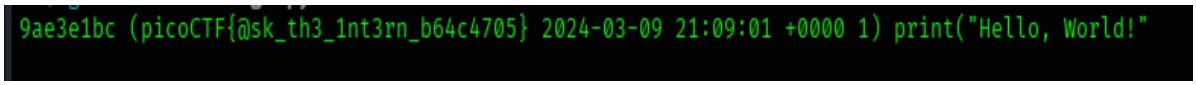
```
┌──(kali㉿kali)-[/media/sf_Downloaded_Cybersecfiles/drop-in]
└─$ git blame message.py
fatal: detected dubious ownership in repository at '/media/sf_Downloaded_Cybersecfiles/drop-in'
To add an exception for this directory, call:

        git config --global --add safe.directory /media/sf_Downloaded_Cybersecfiles/drop-in

┌──(kali㉿kali)-[/media/sf_Downloaded_Cybersecfiles/drop-in]
└─$ git config --global --add safe.directory /media/sf_Downloaded_Cybersecfiles/drop-in
```

(This only needs to be done once per folder.)

6. Ran git blame to see who wrote each line:

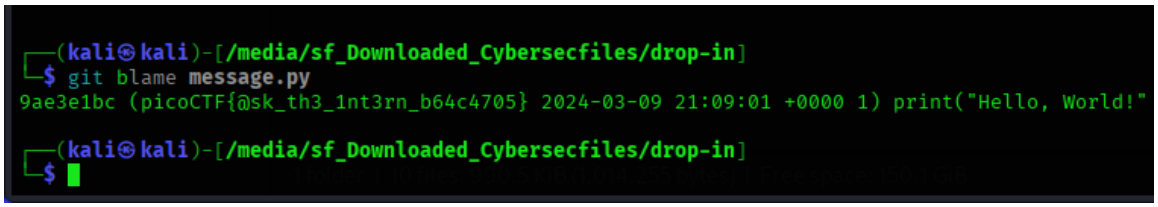
   git blame message.py

7. Found that the **author of the broken line** contained the hidden flag:



V. Result

picoCTF{ picoCTF{@sk_th3_1nt3rn_b64c4705}}



VI. Explanation

- Git keeps track of **every change** made to files.
- git blame <file> shows the **author** of each line.
- In this challenge, the broken Python line was intentionally written by a "fake author" whose name **was actually the flag**.
- We didn't need to run the Python file because the flag was stored in the **metadata (commit author)**, not in the program itself.
- **Important note:** Running the Python file would fail because the code is broken (print("Hello, World!" is missing a closing parenthesis).

VII. Conclusion

This challenge demonstrates the power of **Git for tracing authorship**. Even when files are broken or unsafe to run, you can recover information like flags from the **commit history**.

- Git is an essential tool for version control and debugging.
- Commands like git blame and git show <commit> help identify changes, authors, and hidden metadata.

— **The Analyst: Hyposelenia**