



REDHUNT LABS

DISCOVER. ATTACK. REPEAT.



NULLCON

INTERNATIONAL SECURITY CONFERENCE

Hidden in Plain Sight

Large-Scale Exposure of Dangling Commits on Major Git Platforms

Kumar Ashwin



- Leads Research & Consulting at RedHunt Labs
- Dabbles in **Web, Cloud & Supply Chain Security**
- Speaker/Trainer: **BlackHat, x33fcon, c0c0n, etc.**

@0xCardinal on social platforms

<https://kumarashwin.com>



What's on the Menu?

What are Dangling Commits?

Why is this a Security Risk?

Analyzing the Impact

Large-Scale Data Insights

Fixing & Preventing the Issue

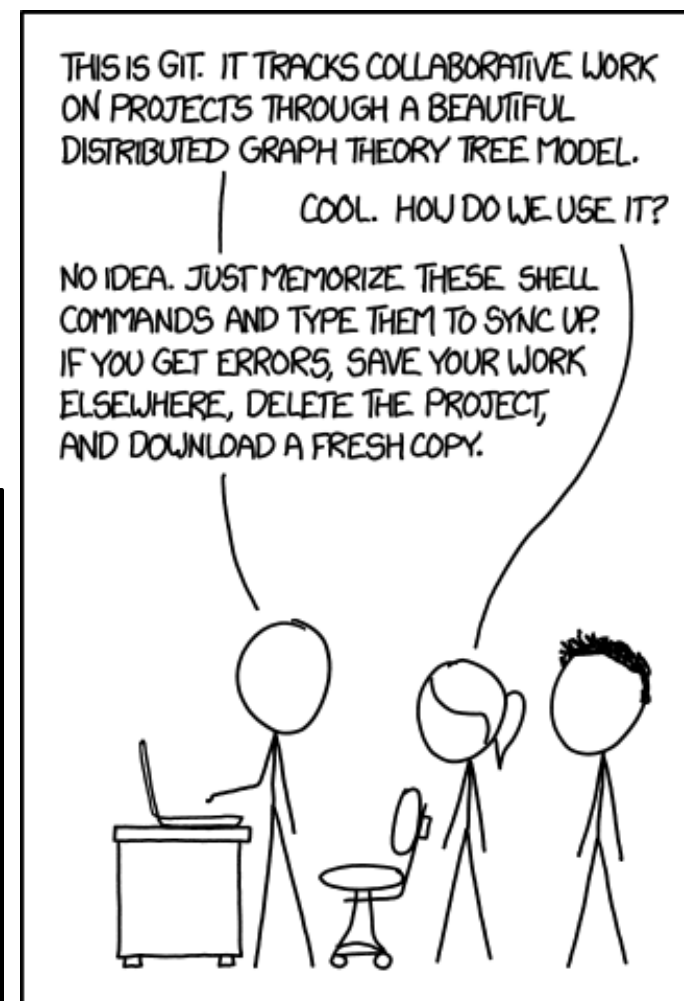
Ending Notes

You know **git**,
right?

& **git commits**?

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.		

<https://xkcd.com/1296/>



How many of you have used git?

How many of you have made commits to a git repository?

How many of you have memorized the git commands?

How many of you have pushed sensitive data to a git repository?



<https://xkcd.com/1597/>



How to Delete Git Commits?

Consensus is...

12Next



Careful: `git reset --hard` **WILL DELETE YOUR WORKING DIRECTORY CHANGES.** Be sure to **stash any local changes you want to keep** before running this command.

5666

Assuming you are sitting on that commit, then this command will wack it...

```
git reset --hard HEAD~1
```





 The `HEAD~1` means the commit before head.

 Or, you could look at the output of `git log`, find the commit id of the commit you want to back up to, and then do this:

```
git reset --hard <sha1-commit-id>
```

If you already pushed it, you will need to do a force push to get rid of it...

```
git push origin HEAD --force
```

<https://stackoverflow.com/questions/1338728/how-do-i-delete-a-commit-from-a-branch>

Deleting a Commit from a Remote Repository

Deleting a commit from a remote repository can be dangerous, as it always requires rewriting the Git history of the target branch. Proceed with caution and communicate with your other collaborators prior to running any destructive commands.

Deleting the most recent commit

If you've pushed a commit to a remote repository and want to remove it:

1. First delete the commit locally by running:

```
git reset --hard HEAD~1
```

This command deletes the most recent commit from your local repository, and **discards the changes** from that commit.

2. Force push to remote:

```
git push origin <branch-name> --force
```

Replace `<branch-name>` with your current branch. This overwrites the remote branch with your local branch,

If you'd like to delete the commits up to a specific commit, run `<git log>` into the command line to find the specific commit id, and then run the following command:

```
git reset --hard <sha1-commit-id>
```

This command discards all working tree changes and moves HEAD to the commit chosen.

Deleting commits already pushed

Alternatively, if your changes are **already pushed**, simply run the following command:

```
git push origin HEAD --force
```

<https://articles.assembla.com/en/articles/2941346-how-to-delete-commits-from-a-branch-in-git>

<https://graphite.dev/guides/how-to-delete-a-git-commit>

What if I told you that deleted commits don't always disappear?

 This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

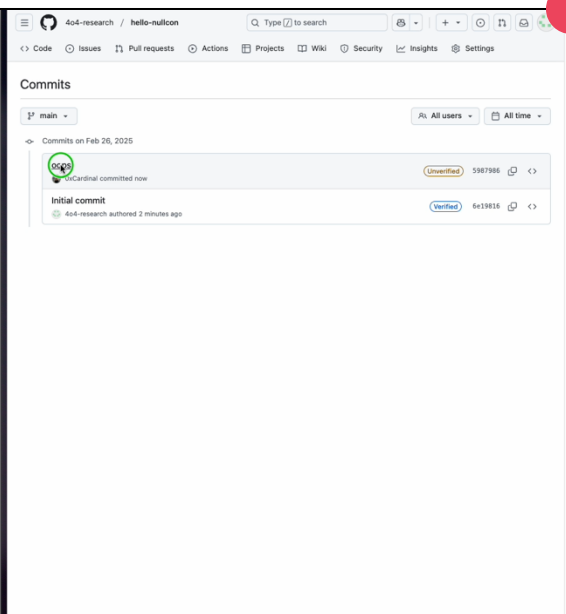
Don't believe me, look for yourself...

```

→ hello-nullcon git:(main) x git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        secret.gif

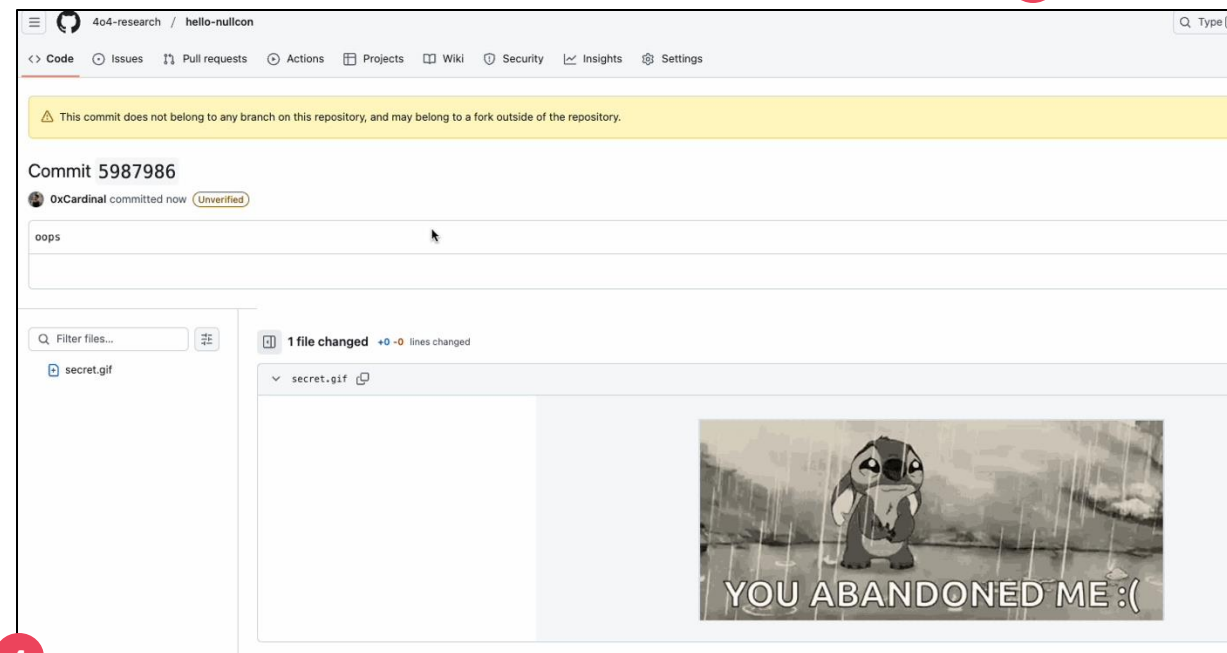
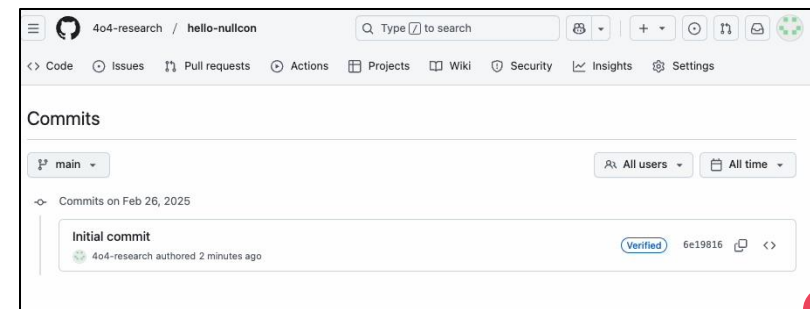
nothing added to commit but untracked files present (use "git add" to track)
→ hello-nullcon git:(main) x git add .
→ hello-nullcon git:(main) x git commit -am "oops"
[main 5987986] oops
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 secret.gif
→ hello-nullcon git:(main) git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 37.57 KiB | 18.78 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To research:404-research/hello-nullcon.git
6e19816..5987986 main -> main
→ hello-nullcon git:(main)
  
```



```

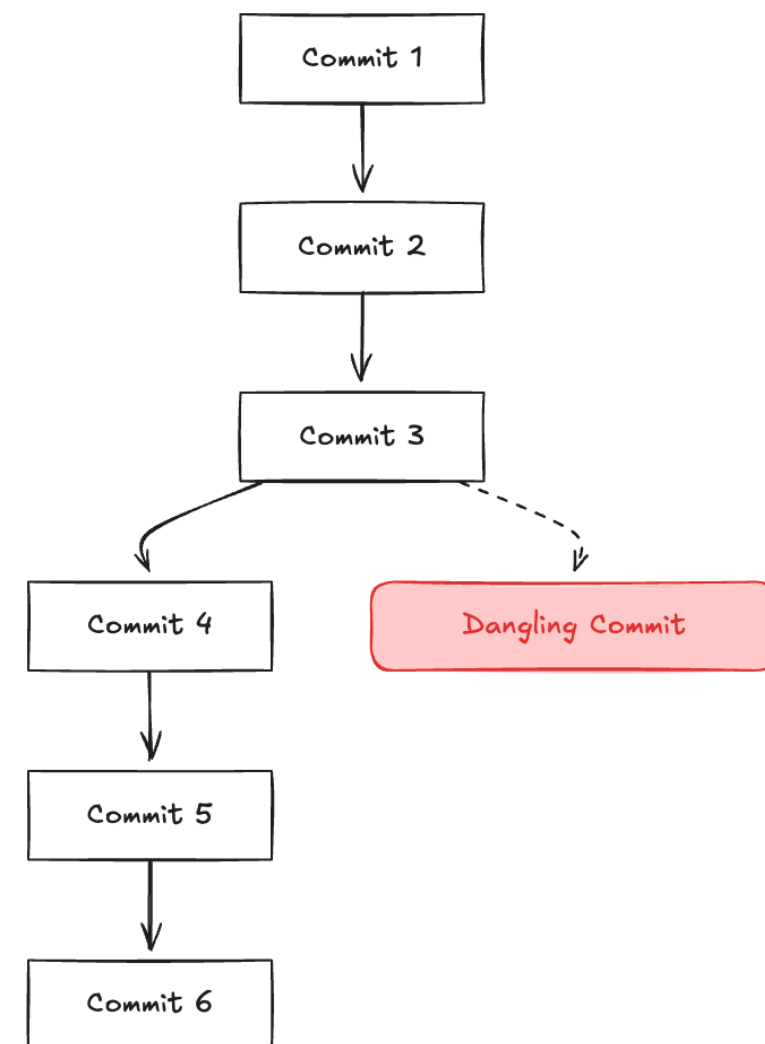
→ hello-nullcon git:(main) git reset HEAD~1 --hard
HEAD is now at 6e19816 Initial commit
→ hello-nullcon git:(main) git push --force
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To research:404-research/hello-nullcon.git
+ 5987986...6e19816 main -> main (forced update)
→ hello-nullcon git:(main)
  
```

4



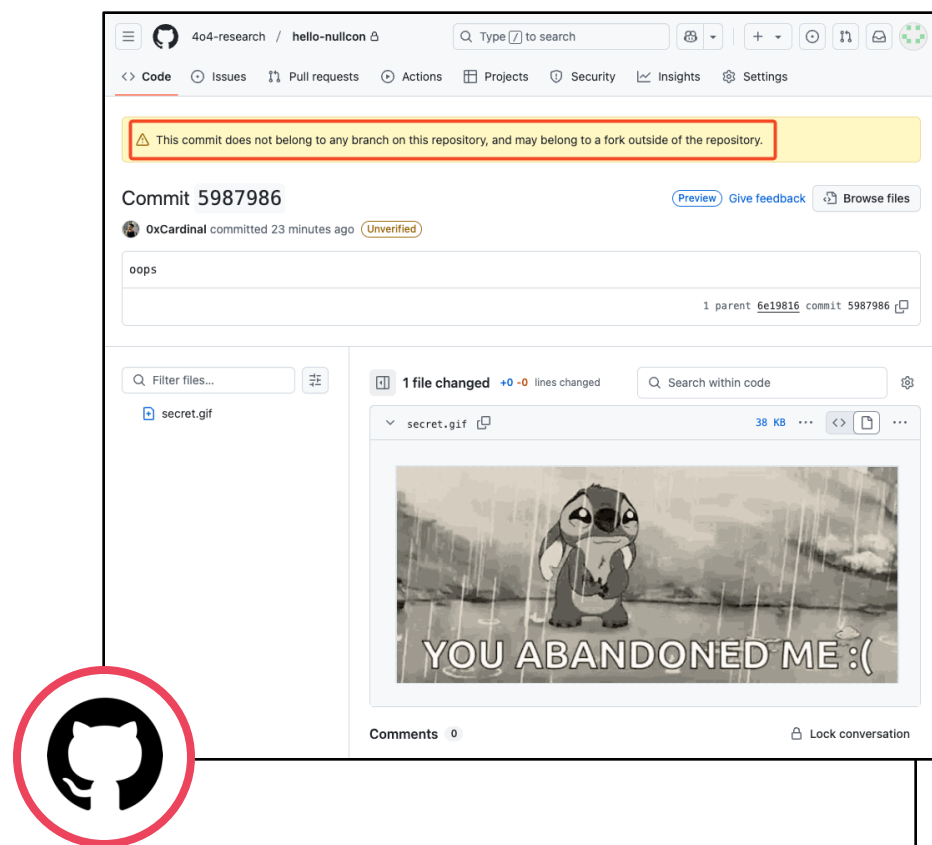
What is a **Dangling Commit**?

- A commit that exists but is **not connected to any branch**.
- Git doesn't delete it immediately—it keeps it... just in case.



Dangling Commits

Across Platforms



```

→ hello-nulicon git:(main) git log --oneline
6e19816 (HEAD -> main, origin/main, origin/HEAD) Initial commit
→ hello-nulicon git:(main) git fsck --lost-found
Checking object directories: 100% (256/256), done.
Checking objects: 100% (3/3), done.
dangling commit 5987986df4d05394155f70d06d36a38e60abc559
  
```

f6d8154

⚠ This commit is not reachable from any branch or tag in this repository. It may be from a fork outside of this repository.

 Oxcardinal committed f6d8154
2024-11-13

add secret

[View source](#) [Approve](#) [Settings](#)

mar Ashwin / Abcd / Commits / f5fa21b1

Commit f5fa21b1 authored 3 months ago by  Oxcardinal

[Browse files](#) [Options](#)

add secret

parent bb2f8243

🔍 No related branches found

🔍 No related tags found

🔍 No related merge requests found

Changes 1

Showing 1 changed file with 1 addition and 0 deletions

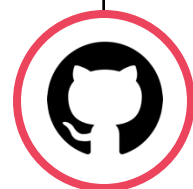
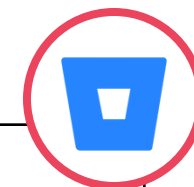
[Hide whitespace changes](#) [Inline](#) [Side-by-side](#)

secret-file 0 → 100644

+1 -0 [View file @ f5fa21b1](#)

1 + something

Please [register](#) or [sign in](#) to comment



Why Is a Dangling Commit an **Issue**?

- When you remove something, you don't want it there...
- Maybe a mistake, or just cleaning things up.
- Or... you're trying to hide something? 😊

But Git doesn't forget – and dangling commits can be a **gold mine** of secrets.



⚠️ This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Commit d[redacted]
[redacted] committed on [redacted]
Fix: gitignore [redacted]

Filter files...

env.dev
env.prod

2 files changed +54 -0 lines changed
env.dev

env.prod

```

... @@ -0,0 +1,27 @@
1 + # RDBMS
2 + MYSQL_HOST=mariadb[redacted].rds.amazonaws.com
3 + MYSQL_PORT=3306
4 + MYSQL_USER=[redacted]ova
5 + MYSQL_PASSWORD=te[redacted]
6 + MYSQL_DATABASE_NAME=[redacted]_bct
7 + MYSQL_LOGGING=true
8 +
9 + # AWS S3
10 + AWS_BUCKET_NAME=re[redacted]
11 + AWS_BUCKET_REGION=ap-northeast-2
12 + AWS_ACCESS_KEY_ID=AKIA3[redacted]
13 + AWS_SECRET_ACCESS_KEY=z1D4GV[redacted]
14 + S3_URL=https://reptim[redacted]
15 +
16 + # Redis
17 + REDIS_HOST=www.reptimate.store
18 + REDIS_PASSWORD=tea[redacted]
19 +
20 + # JWT
21 + JWT_SECRET=fpqxla[redacted]
22 +
23 + # Slack
24 +
25 + # nodemailer
26 + NODE_MAILER_ID=iot[redacted]
27 + NODE_MAILER_PASSWORD=dfu[redacted]

```

We reached out to Git platforms about this issue.

They have **no plans to fix it but shared guidance on manual removal.**

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository>

Cool, we have a solution now, let's try that!

```
→ hello-again-nullcon git:(main) git-filter-repo --sensitive-data-removal --invert-paths --path secret.env --force
NOTICE: Fetching all refs from origin to make sure we rewrite
        all history that may reference the sensitive data, via
        git fetch -q --prune --update-head-ok --refmap "" origin +refs/*:refs/*
Parsed 2 commits
New history written in 2.80 seconds; now repacking/cleaning...
You rewrote 1 (of 2) commits.

NOTE: First Changed Commit(s) is/are:
      4e4186057f53818fa8fe3c3a1ca9573a1bc4ca79
NOTE: LFS object orphaning not checked (LFS not in use)

Repacking your repo and cleaning out old unneeded objects
HEAD is now at e6a1432 Initial Commit
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Completely finished after 2.90 seconds.
```

P.S. - **BFG Repo-Cleaner** can also be used.

```
→ hello-again-nullcon git:(main) git push --force --mirror origin

Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 234 bytes | 234.00 KiB/s, done.
Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
To research:404-research/hello-again-nullcon.git
+ 4e41860...e6a1432 main -> main (forced update)
```

4o4-research / hello-again-nullcon

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security 1

Insights

Settings

⚠️

This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Commit 4e41860

[Preview](#)
[Give feedback](#)
[Browse files](#)

4o4-research

committed 4 minutes ago

not committing secret

1 parent [e6a1432](#) commit 4e41860

secret.env

1 file changed

+36 -0 lines changed

secret.env

+36

```

... @@ -0,0 +1,36 @@
1 + # 🚨 WARNING: Definitely NOT real secrets. Move along, hackers. 🚨
2 + APP_ENV=production
3 + DEBUG=False # Because we totally don't have bugs 🐛
4 +
5 + # Database Credentials (Totally Secure™)
6 + DB_HOST=localhost # Because putting production DB in the cloud is too mainstream

```

HOPE YOU ENJOYED MY PRESENTATION

THANK YOU FOR LISTENING

How do we **remove the dangling commit** then?

✅ Option 1: Contact GitHub Support

💬 Submit a request and hope for the best!

✅ Option 2: Delete the Repository

🗑️ If it's gone, it's gone... right? (Maybe... 😬)

✅ Option 3: Make the Repository Private

🔒 Out of sight, out of mind?

Fully removing the data from GitHub [🔗](#)

After using `git-filter-repo` to remove the sensitive data and pushing your changes to GitHub, you must take a few more steps to fully remove the data from GitHub.

- 1 Contact us through the [GitHub Support portal](#), and provide the following information:
 - The owner and repository name in question (e.g. YOUR-USERNAME/YOUR-REPOSITORY).
 - The number of affected pull requests, found in the previous step. This is used by Support to verify you understand how much will be affected.
 - The First Changed Commit(s) reported by `git-filter-repo` (Look for `NOTE: First Changed Commit(s)` in its output.)
 - If `NOTE: There were LFS Objects Orphaned by this rewrite` appears in the `git-filter-repo` output (right after the First Changed Commit), then mention you had LFS Objects Orphaned and upload the named file to the ticket as well.

If you have successfully cleaned up all references other than PRs, and no forks have references to the sensitive data, Support will then:

- Dereference or delete any affected PRs on GitHub.
- Run a garbage collection on the server to expunge the sensitive data from storage.
- Remove cached views.
- If LFS Objects are involved, delete and/or purge the orphaned LFS objects.

📌 Important

GitHub Support won't remove non-sensitive data, and will only assist in the removal of sensitive data in cases where we determine that the risk can't be mitigated by rotating affected credentials.

- 2 Collaborators must [rebase](#), not merge, any branches they created off of your old (tainted) repository history. One merge commit could reintroduce some or all of the tainted history that you just went to the trouble of purging. They may need to take additional steps as well; see [Make sure other copies are cleaned up: clones of colleagues](#) in the `git-filter-repo` manual.

But how will you get these dangling commits? If you can't find it, there is NO impact.

Let me tell you a secret...we were also able to enumerate these dangling commits using...

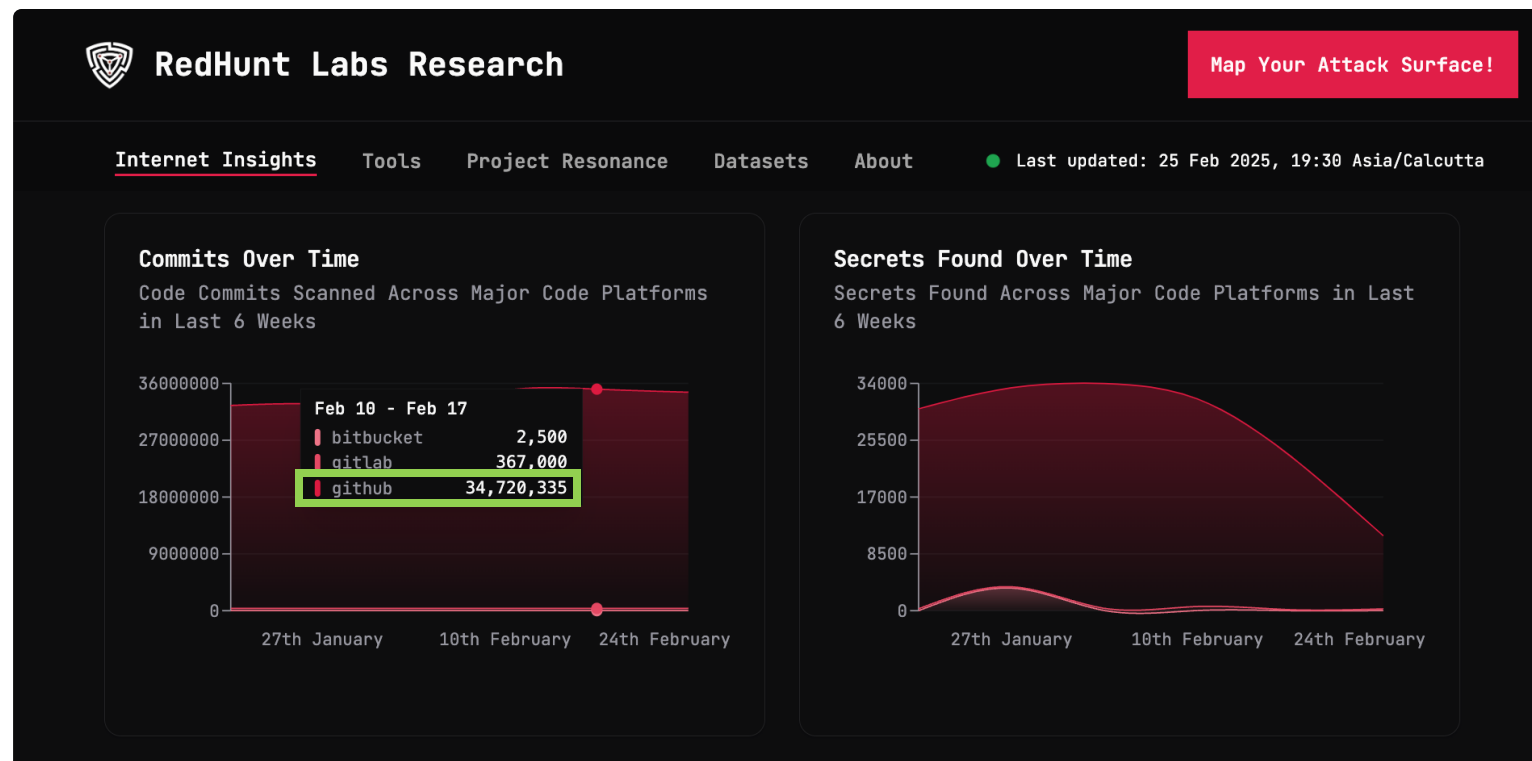
Events API 🎉

Why GitHub? Understanding Our Dataset

~ 5 years

Sample Data Timeline

All insights are drawn specifically from **GitHub commit events**.



<https://research.redhuntlabs.com>

GitHub Events

```
{
  "id": "46979116854",
  "type": "PushEvent",
  "actor": {
    "id": 188773812,
    "login": "4o4-research",
    "display_login": "4o4-research",
    "gravatar_id": "",
    "url": "https://api.github.com/users/4o4-research",
    "avatar_url": "https://avatars.githubusercontent.com/u/188773812?"
  },
  "repo": {
    "id": 939329639,
    "name": "4o4-research/hello-again-nullcon",
    "url": "https://api.github.com/repos/4o4-research/hello-again-nullcon"
  },
  "payload": {
    "repository_id": 939329639,
    "push_id": 22869189109,
    "size": 0,
    "distinct_size": 0,
    "ref": "refs/heads/main",
    "head": "69cca5b86242799d3dbcfc53bc7ea5a9a2b9154a",
    "before": "a8c2afe8e86147366c120be2a01fd3a1334e5635",
    "commits": []
  },
  "public": true,
  "created_at": "2025-02-26T12:28:47Z"
},
```

<https://api.github.com/repos/4o4-research/hello-again-nullcon/events>

The dangling commit IDs can be enumerated from the GitHub Events API –

- **Repository Events**
- **User Events**
- **Public Events**

Let's Talk Patterns – Key Observations

- **Force push** is often used to hide these events.
- **Push events with no commits** are a strong indicator.
- But... not all matching patterns are dangling commits!
- Such an event occurs during multiple git actions -
 - Force Push with No New Commits
 - Branch Deletion
 - Tag Creation or Deletion
 - Empty Push
 - Repository Initialization with No Files
 - Protected Branch with Push Rejected
 - Merge a Pull Request?

```
{
  "id": "98765432101",
  "type": "PushEvent",
  "actor": {
    "id": 56789012,
    "login": "cyber_ninja",
    "display_login": "cyber_ninja",
    "gravatar_id": "",
    "url": "https://api.github.com/users/cyber_ninja",
    "avatar_url": "https://avatars.githubusercontent.com/u/56789012?"
  },
  "repo": {
    "id": 1122334455,
    "name": "cyber_ninja/stealth_project",
    "url": "https://api.github.com/repos/cyber_ninja/stealth_project"
  },
  "payload": {
    "repository_id": 1122334455,
    "push_id": 87654321098,
    "size": 0,
    "distinct_size": 0,
    "ref": "refs/heads/main",
    "head": "abcdef1234567890abcdef1234567890abcdef12",
    "before": "123456abcdef7890123456abcdef7890123456ab",
    "commits": []
  },
  "public": true,
  "created_at": "2025-02-26T10:30:45Z"
}
```

How Do We Filter the Noise?

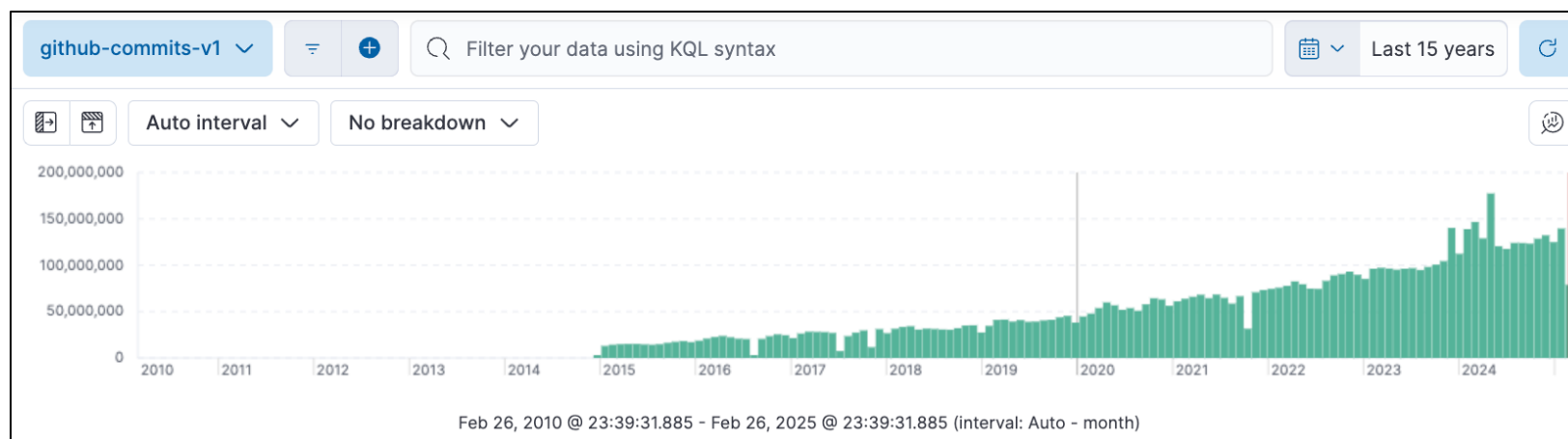
- We analysed GitHub's **branch_commits** endpoint.
- https://github.com/USER/REPO/branch_commits/COMMIT_ID
- If a commit isn't linked to any branch → It's dangling!
- This is the same method GitHub uses internally.

96	https://github.com	POST	/commits/badges	✓	200
95	https://github.com	GET	/404-research/hello-again-nullcon/security/overall-count		200
94	https://github.com	GET	/404-research/hello-again-nullcon/branch_commits/a8c2afe8e86147366c120be2a...		200
93	https://collector.github.com	POST	/github/collect	✓	204
92	https://api.github.com	POST	/_private/browser/stats	✓	200
			search/hello-again-nullcon/commit/a8c2afe		200
			/browser/stats	✓	200

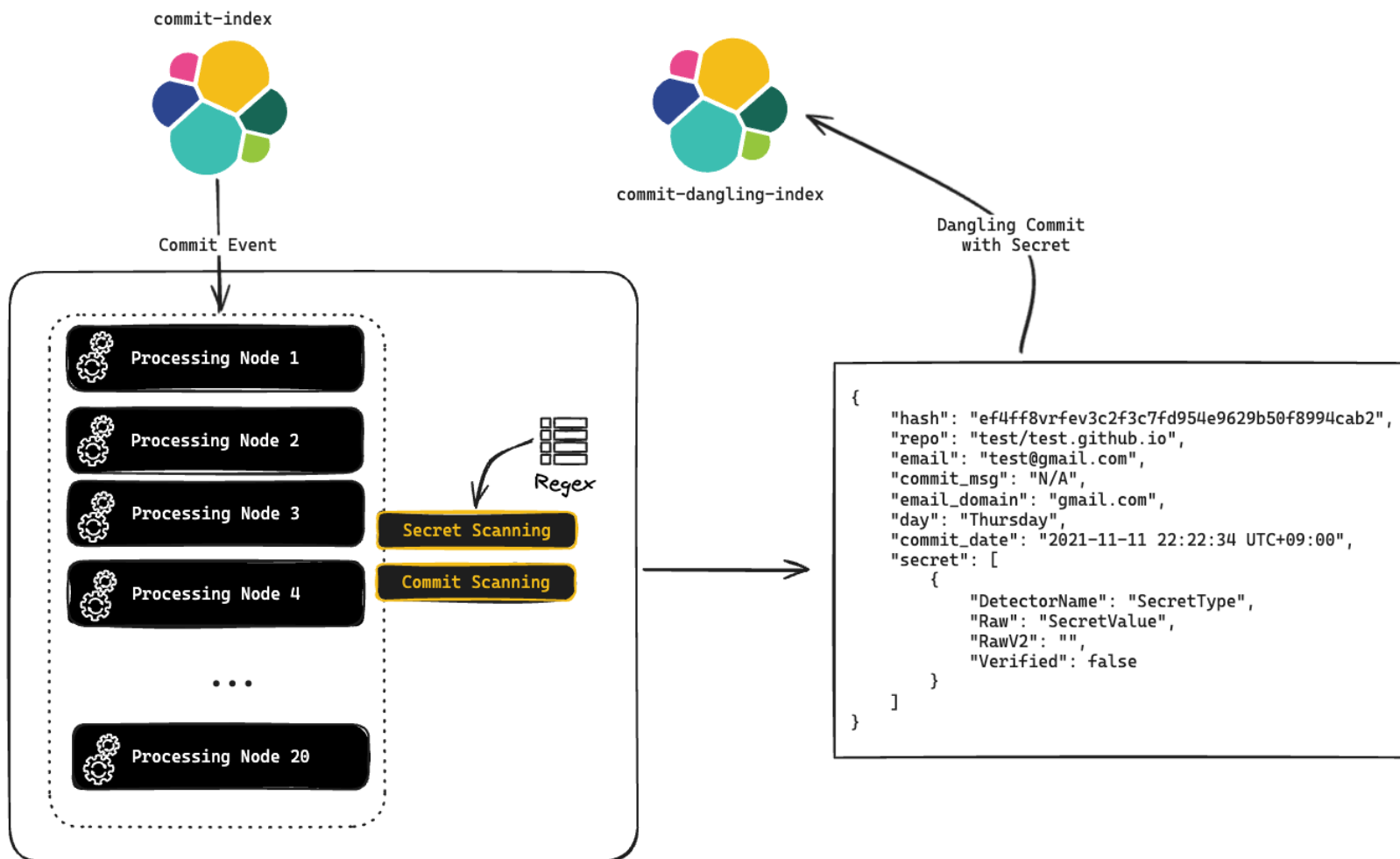
Request	Response
<pre> 1 GET /404-research/hello-again-nullcon/branch_commits/a8c2afe8e86147 366c120be2a01fd3a1334e5635 HTTP/2 2 Host: github.com 3 Cookie: _octo=GH1.1.480683446.1731085208; _device_id= a1925968724b475410e2a91958e12e27; logged_in=no; cpu_bucket=lg; preferred_color_mode=dark; tz=Asia%2FCalcutta; _gh_sess= yBTjc802j4URYtCHbKsMviCU%2FD2cpHI5tcAsoyPLRUVjDcQupKfbZIRWEFDEg PH%2Bh3vHwrdT37M0p8bHCLes%2Fz9z4UF8eh0Ro12nRd0MIRgWly9GnLQa49G q2a%2Fqf83yPVP94sqwynMil%2B%2BSo647VrBWrIaIoc3oR2Xa%2BiaF0cp2N 6bE9s0Ie0SX1ovE70sfl%2Bm6Ikzpl1IZ1eYza%2BfN6zrgj22uuVfibBsCSX APZis0DpmMdcuiGgbrFcEmfRm5UYV1Rzk16WwGsIpzoQ%3D%3D---9elrrWImqbh GW14g---0ED0oMz9%2FmxmDUTGJutMMw%3D%3D 4 Sec-Ch-Ua-Platform: "macOS" 5 X-Requested-With: XMLHttpRequest 6 Accept-Language: en-GB,en;q=0.9 7 Accept: text/html 8 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99" 9 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 </pre>	<pre> 1 user-images.githubusercontent.com/ secured-user-images.githubusercontent.com/ private-user-images.githubusercontent.com github-production-user-asset-6210df.s3.amazonaws.com gist.githubusercontent.com; script-src github.githubassets.com; style-src 'unsafe-inline' github.githubassets.com; upgrade-insecure-requests; worker-src github.com/assets-cdn/worker/ github.com/webpack/ github.com/assets/ gist.github.com/assets-cdn/worker/ 14 Accept-Ranges: bytes 15 X-HTML-Safe: 3fdb2557d1c5195f19e4eca921f6acf26ea58656f95d2c14ccd35ea9c31e8426 16 Content-Length: 73 17 X-GitHub-Request-Id: 1BEA:F0006:161F4F:1AB21D:67BF56AA 18 19 20 21 22 </pre>

The Missing Ingredient: Lots of GitHub Events

- GitHub **only retains event history for 30 days**
- To **truly measure the impact**, we need **long-term data**
- We needed **more than just 30 days...** we needed **1500+ days** as our sample data.
- At RedHunt Labs, we have maintained a **historical dataset of commits spanning 10+ years** and this data became invaluable in uncovering the real scope of dangling commits.

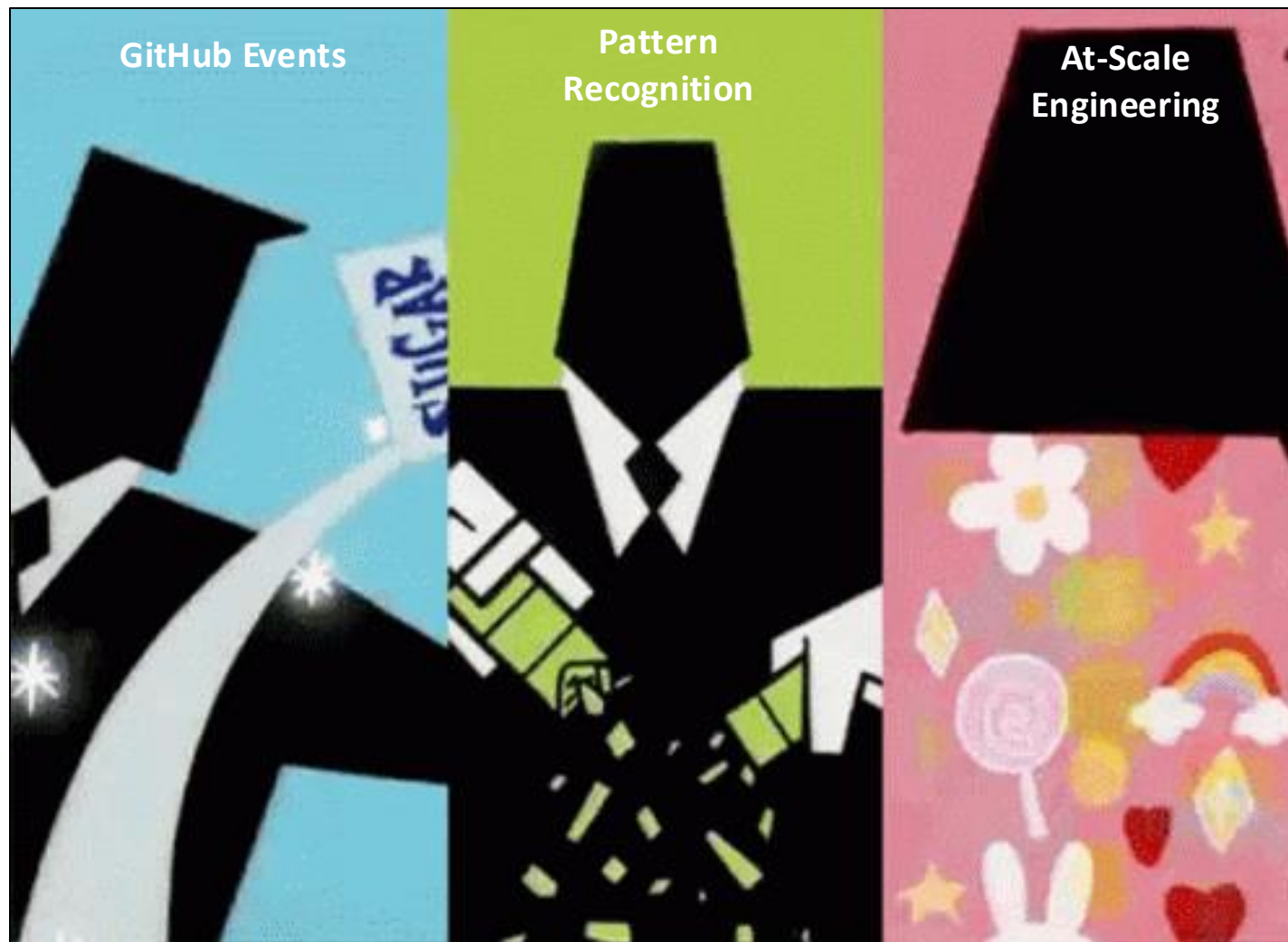


Some Engineering ...



... and some Jugaad

<pre>ubuntu@ip-172-31-7-169:~/output\$ cd .. ubuntu@ip-172-31-7-169:~\$ ls -l patch_files/ wc -l 16767 16767 ubuntu@ip-172-31-7-169:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 7.0G 186G 4% / 17867 ubuntu@ip-172-31-7-169:~\$</pre>	<pre>ubuntu@ip-172-31-12-222:~/output\$ cd .. ubuntu@ip-172-31-12-222:~\$ ls -l patch_files/ wc -l 14588 14588 ubuntu@ip-172-31-12-222:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.9G 188G 3% / 15414 ubuntu@ip-172-31-12-222:~\$</pre>	<pre>ubuntu@ip-172-31-14-6:~/output\$ cd .. ubuntu@ip-172-31-14-6:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 15527 15527 ubuntu@ip-172-31-14-6:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.8G 187G 3% / 16614 ubuntu@ip-172-31-14-6:~\$</pre>	<pre>ubuntu@ip-172-31-4-15:~/output\$ cd .. ubuntu@ip-172-31-4-15:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 9586 9586 ubuntu@ip-172-31-4-15:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.2G 188G 3% / 18187 ubuntu@ip-172-31-4-15:~\$</pre>
<pre>ubuntu@ip-172-31-5-242:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 9054 9054 ubuntu@ip-172-31-5-242:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.4G 189G 3% / 9476 ubuntu@ip-172-31-5-242:~\$</pre>	<pre>ubuntu@ip-172-31-11-122:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 18861 18862 ubuntu@ip-172-31-11-122:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.4G 189G 3% / 19948 ubuntu@ip-172-31-11-122:~\$</pre>	<pre>ubuntu@ip-172-31-0-103:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 14161 14161 ubuntu@ip-172-31-0-103:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.8G 187G 4% / 14988 ubuntu@ip-172-31-0-103:~\$</pre>	<pre>ubuntu@ip-172-31-3-154:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 10167 10167 ubuntu@ip-172-31-3-154:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.4G 188G 3% / 10786 ubuntu@ip-172-31-3-154:~\$</pre>
<pre>ubuntu@ip-172-31-10-121:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 12265 12265 ubuntu@ip-172-31-10-121:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.7G 189G 3% / 12980 ubuntu@ip-172-31-10-121:~\$</pre>	<pre>ubuntu@ip-172-31-9-252:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 18982 18982 ubuntu@ip-172-31-9-252:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.2G 189G 3% / 20064 ubuntu@ip-172-31-9-252:~\$</pre>	<pre>ubuntu@ip-172-31-5-218:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 14874 14874 ubuntu@ip-172-31-5-218:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.1G 188G 3% / 15766 ubuntu@ip-172-31-5-218:~\$</pre>	<pre>ubuntu@ip-172-31-13-226:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 11977 11977 ubuntu@ip-172-31-13-226:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.7G 189G 3% / 12677 ubuntu@ip-172-31-13-226:~\$</pre>
<pre>ubuntu@ip-172-31-4-93:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 11874 11874 ubuntu@ip-172-31-4-93:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.4G 188G 3% / 12561 ubuntu@ip-172-31-4-93:~\$</pre>	<pre>ubuntu@ip-172-31-1-222:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 13524 13524 ubuntu@ip-172-31-1-222:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.2G 188G 3% / 14319 ubuntu@ip-172-31-1-222:~\$</pre>	<pre>ubuntu@ip-172-31-10-65:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 18510 18510 ubuntu@ip-172-31-10-65:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.3G 189G 3% / 19593 ubuntu@ip-172-31-10-65:~\$</pre>	<pre>ubuntu@ip-172-31-12-125:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 11935 11936 ubuntu@ip-172-31-12-125:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.4G 189G 3% / 12528 ubuntu@ip-172-31-12-125:~\$</pre>
<pre>ubuntu@ip-172-31-2-23:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 10543 10543 ubuntu@ip-172-31-2-23:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.7G 188G 3% / 11170 ubuntu@ip-172-31-2-23:~\$</pre>	<pre>ubuntu@ip-172-31-1-23:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 11407 11408 ubuntu@ip-172-31-1-23:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.5G 188G 3% / 11936 ubuntu@ip-172-31-1-23:~\$</pre>	<pre>ubuntu@ip-172-31-14-16:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 13714 13714 ubuntu@ip-172-31-14-16:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 5.3G 188G 3% / 14513 ubuntu@ip-172-31-14-16:~\$</pre>	<pre>ubuntu@ip-172-31-12-220:~\$ ls -l patch_files/ wc -l; ls -l output/ wc -l 10774 10774 ubuntu@ip-172-31-12-220:~\$ df -h head -n 2 ; ls -l patch_files/ wc -l Filesystem Size Used Avail Use% Mounted on /dev/root 193G 4.9G 188G 3% / 11392 ubuntu@ip-172-31-12-220:~\$ █</pre>



These were the ingredients chosen to create the perfect dangling commit detection at scale!

We uncovered the real impact of dangling commits! 🚀

~ 5,400,000,000

GitHub Commits

~ **66,000,000**

Dangling GitHub Commits

~ **550,000**

**Secrets Found in Dangling
Commits**

???

**What day of the week sees the
most dangling commits?**

Friday

What day of the week sees the most dangling commits?

Top 10 values of day	% Secrets in Dangling Commits
Friday	22%
Monday	20%
Wednesday	18%
Tuesday	15%
Thursday	15%
Sunday	6%
Saturday	5%

snyk.io

naver.com

qq.com

checkmarx.com

163.com

redhat.com

Top Companies Responsible for the Most Dangling Commits

```
git commit -am "Saving release notes"
```

```
git commit -am "---"
```

```
git commit -am "N/A"
```

```
git commit -am "release: 4.0.0"
```

```
git commit -am "auto pull"
```

```
git commit -am "added chatgpt data"
```

```
git commit -am "test"
```

Top Commit Messages for Dangling Commits

Top **Types of Secrets** Found



The State of Dangling Commits & Secrets Exposure

~ **5,400,000,000**

GitHub Commits

~ **66,000,000**

Dangling GitHub Commits

~ **550,000**

Secrets Found in Dangling Commits

Friday

What **day of the week** sees the **most dangling commits**?

```
git commit -am "Saving release notes"
git commit -am "---"
git commit -am "N/A"
git commit -am "release: 4.0.0"
git commit -am "auto pull"
git commit -am "added chatgpt data"
git commit -am "test"
```

Top Commit Messages for Dangling Commits

snyk.io

naver.com

qq.com

checkmarx.com

163.com

redhat.com

Top Companies Responsible for the Most Dangling Commits

Top Types of Secrets Found





How Do We **Avoid Creating Dangling Commits?**

Technical Solution

Host Your Own Git Management System

- Provides full control over commit policies.
- Costly and complex to scale.
- Risk persists outside your controlled repositories – public, forks, repositories.

Automated Detection & Monitoring

- Use detection logic & scanners to flag issues.
- Monitor commit history for force-pushes and deletions.

Immediate Secret Rotation

- If a secret key is pushed, **rotate it immediately** to prevent exposure.

How Do We **Avoid Creating Dangling Commits?**

Human Factors & Awareness

Developer Awareness & Training

- Educate developers on the risks of dangling commits.
- Implement secure commit practices in workflows.

Encouraging Secure Commit Hygiene

- Adopt best practices in both **open-source** and **enterprise** projects.
- Enforce pre-commit hooks, signed commits, and branch protection policies.

Thanks!

For any queries, reach out to research@redhuntlabs.com

@0xCardinal on social platforms <https://kumarashwin.com>