



Stop Committing Your Secrets - Git Hooks To The Rescue

Conf42: DevSecOps 2022

CONF42



Hi, I'm Dwayne



Dwayne McDaniel

- I live in Chicago
- I've been a Developer Advocate since 2016
- On Twitter @mcdwayne
- Happy to chat about anything, hit me up
- Besides tech, I love improv, karaoke and going to rock and roll shows!



About GitGuardian



GitGuardian is the code security platform for the DevOps generation.

With automated secrets detection and remediation, our platform enables Dev, Sec, and Ops to advance together towards the Secure Software Development Lifecycle.



A Few Incidents

Uber

- Reported: 15 Sept, 2022
- Teenager from the Lapsus\$ hacking group phished login info from a super admin
- Immediately discovered access credentials hardcoded in PowerShell scripts that allowed pwnage
- Reported first in the New York Times



A Few Incidents

Toyota

- Reported: 7 October, 2022
- A subcontractor hired to work on the Toyota T-Connect source code pushed a private codebase into a public GitHub repo.
- The repo contained access credentials for a data server, which exposed the emails of 296,019 customers
- The repo was public from December 2017 to September 2022 - **5 years!**



A Few Incidents

Samsung

- Reported: 7 March 2022 and 2 September 2022
- 160GB of data stolen by Lapsus\$ hacking group and published in March, including Galaxy source code containing over 6,000 secrets (API keys, passwords, credentials)
- From July to August customer data was stolen
- No reporting of how many individuals were impacted, nor details on how the threat actors gained access



A Few Incidents

AstraZeneca

- Reported: 3 November 2022
- Developer hardcoded credentials and pushed to GitHub in 2021, giving access to test environments
- "User error" caused an undisclosed amount of patient data to be available in a test environment
- Credentials were exposed for over a year







Master Locksmiths Association (MLA) @MLA_lock... · Jan 17, 2015

Good security fail spotted by one of our members :-)



```
1 package main
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 func main() {
9     databaseName := "53CR3TD4T4B453"
10    secretKey := "SUP3R53CR3T"
11    secretPhrase := "Always know where your towel is. - Douglas Adams, The Hitchhiker's Guide to the Galaxy"
12
13    var dbName string
14    var dbPass string
15
16    fmt.Println("Please enter database name:")
17    fmt.Scanf("%s", &dbName)
18
19    fmt.Println("Please enter database password:")
20    fmt.Scanf("%s", &dbPass)
21
22    if dbName == databaseName && dbPass == secretKey {
23        fmt.Println("Welcome to the database!")
24        fmt.Println("Your secret phrase is: ", secretPhrase)
25        os.Exit(0)
26    }
27    fmt.Println("Sorry, wrong database name or password")
28 }
29
```



In the 2022 edition of The State of Secrets Sprawl

6M secrets found exposed

in 2021 in public GitHub repositories

More than 2x increase compared to 2 Million in 2020!

On average, 3 commits out of 1,000 exposed at least one secret

+50% compared to 2020

<https://www.gitguardian.com/state-of-secrets-sprawl-report-2022>



Who Is Responsible?



Security Is Everyone's Job, At Every Step In The SDLC, Not Just The Security Teams' Responsibility

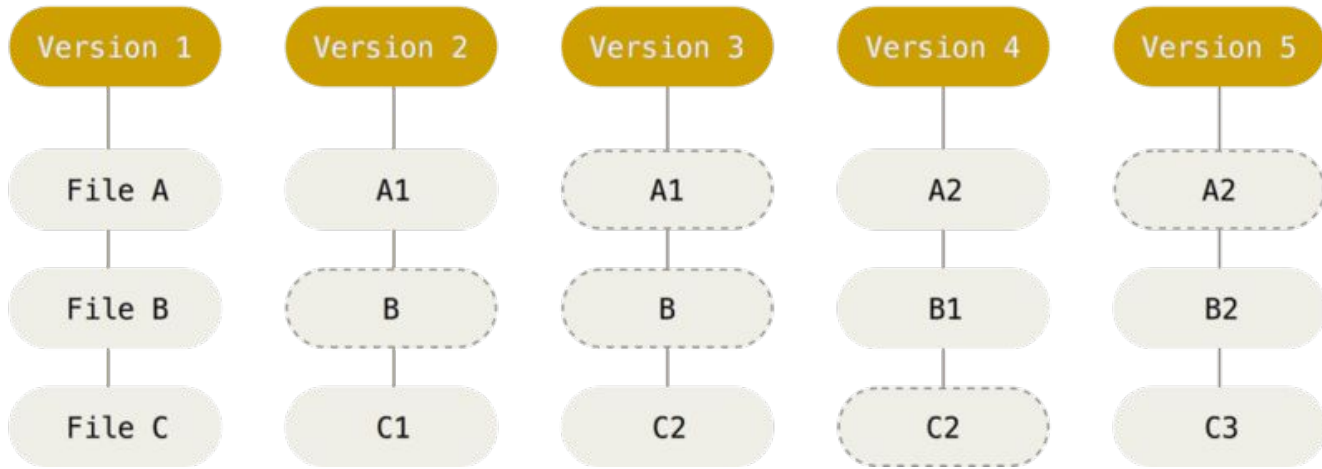


*In the best organizations
developers outnumber
security team members 100:1*

- Alex Rice, HackerOne
#Security@2022



Checkins Over Time



**Git Is At The Heart Of DevOps
And Is The Tool All* Devs Use**



Git Is Awesome!!!

Git does not make code more or less secure

```
GIT(1)
```

```
Git Manual
```

```
NAME
```

```
git – the stupid content tracker
```



Git does give us a way to exclude entire types of files or directories...

.gitignore

```
1 *.gem
2 *.rbc
3 /.config
4 /coverage/
5 /InstalledFiles
6 /pkg/
7 /spec/reports/
8 /spec/examples.txt
9 /test/tmp/
0 /test/version_tmp/
1 /tmp/
2
3 # Used by dotenv library to load environment variables.
4 # .env
5
6 # Ignore Byebug command history file.
7 .byebug_history
8
9 ## Specific to RubyMotion:
0 .dat*
1 .repl_history
2 build/
3 *.bridgesupport
4 build-iPhoneOS/
5 build-iPhoneSimulator/
```



Using a `.gitignore` file can make sure you do not commit a `secrets.json`, an `aws` directory or wherever you store API keys, usernames and passwords, or security certificates**

```
GITIGNORE(5)
```

```
Git Manual
```

```
NAME
```

```
gitignore – Specifies intentionally untracked files to ignore
```



Combine `.gitignore` with secrets managers like Hashicorp Vault or Azure Key Vault and you have eliminated hardcoded secrets leaks...



In a perfect world, that would be the end of the talk.

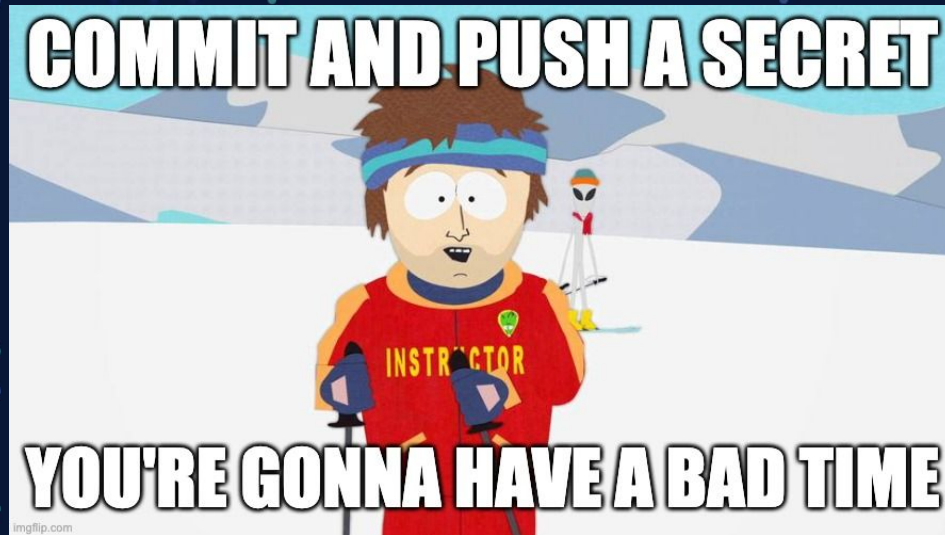
However...

```
1 package main
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 func main() {
9     databaseName := "53CR3TD4T4B453"
10    secretKey := "5UP3R53CR3T"
11    secretPhrase := "Always know where your towel is. - Douglas Adams, The Hitchhiker's Guide to the Galaxy"
12
13    var dbName string
14    var dbPass string
15
16    fmt.Println("Please enter database name:")
17    fmt.Scanf("%s", &dbName)
18
19    fmt.Println("Please enter database password:")
20    fmt.Scanf("%s", &dbPass)
21
22    if dbName == databaseName && dbPass == secretKey {
23        fmt.Println("Welcome to the database!")
24        fmt.Println("Your secret phrase is: ", secretPhrase)
25        os.Exit(0)
26    }
27    fmt.Println("Sorry, wrong database name or password")
28 }
29
```



The issue is not that you tested a secret.

The issue is you forgot to remove it from your code before you committed and pushed.



**You *can* (in theory)
remove secrets
from a shared repo, but
it is not easy**

It's downright painful



"Are you sure you got it out from all the commits and branches?"

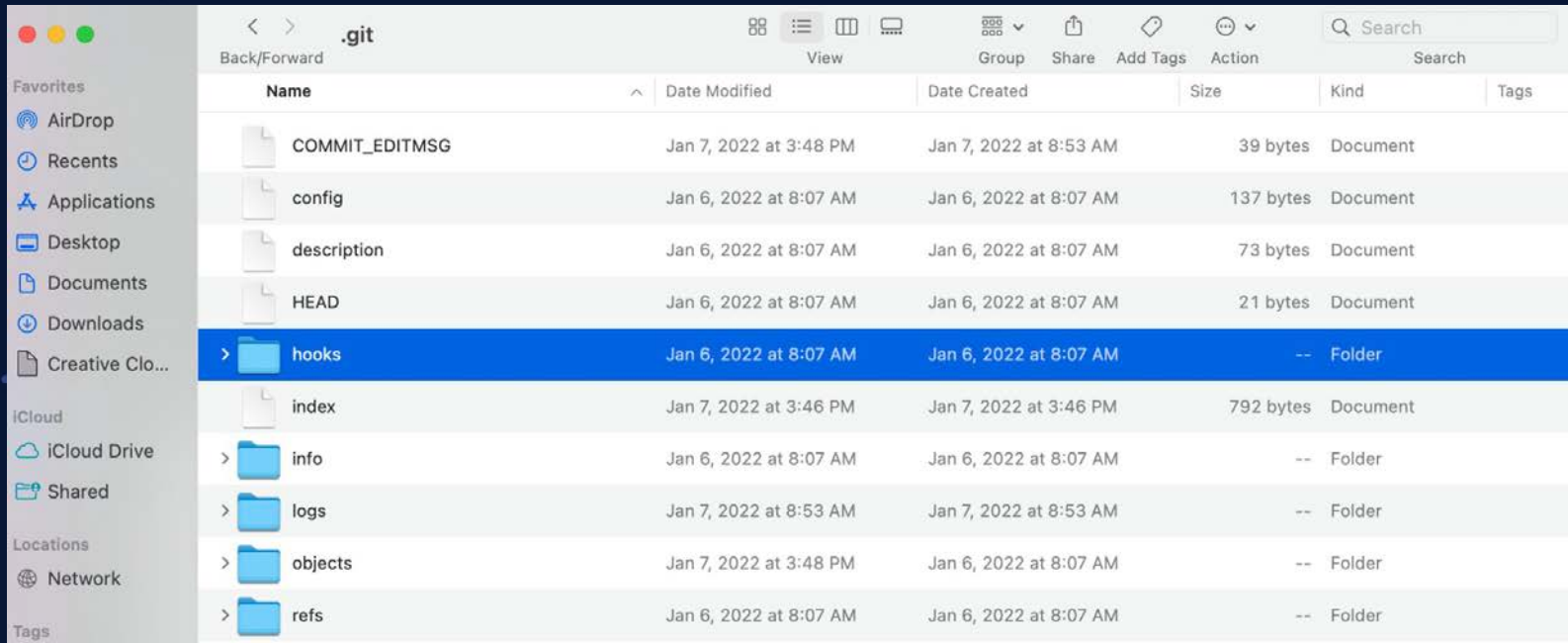


What we need is
some sort of
automation that
stops us from
committing our
secrets...



Git provides a way...

Git Hooks

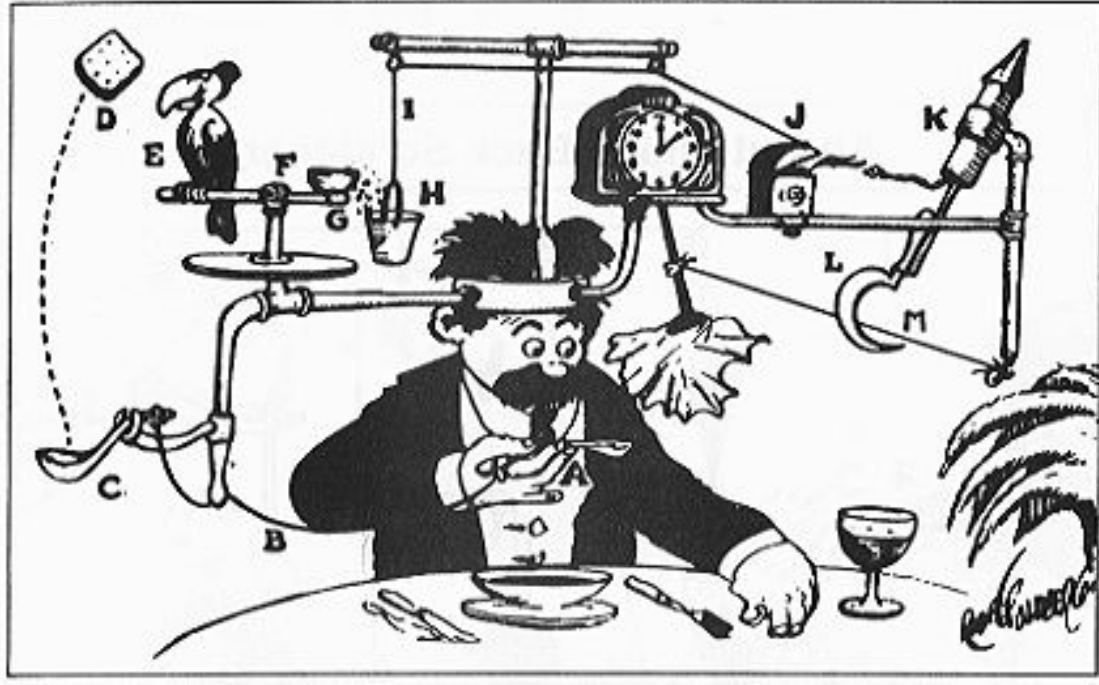


A screenshot of a macOS Finder window showing the contents of a `.git` directory. The window title is `.git`. The sidebar on the left shows various system locations like Favorites, iCloud, and Locations. The main pane displays a table of files and folders within the `.git` directory. The `hooks` folder is highlighted in blue.

| Name | Date Modified | Date Created | Size | Kind | Tags |
|----------------|------------------------|------------------------|-----------|----------|------|
| COMMIT_EDITMSG | Jan 7, 2022 at 3:48 PM | Jan 7, 2022 at 8:53 AM | 39 bytes | Document | |
| config | Jan 6, 2022 at 8:07 AM | Jan 6, 2022 at 8:07 AM | 137 bytes | Document | |
| description | Jan 6, 2022 at 8:07 AM | Jan 6, 2022 at 8:07 AM | 73 bytes | Document | |
| HEAD | Jan 6, 2022 at 8:07 AM | Jan 6, 2022 at 8:07 AM | 21 bytes | Document | |
| > hooks | Jan 6, 2022 at 8:07 AM | Jan 6, 2022 at 8:07 AM | -- | Folder | |
| index | Jan 7, 2022 at 3:46 PM | Jan 7, 2022 at 3:46 PM | 792 bytes | Document | |
| > info | Jan 6, 2022 at 8:07 AM | Jan 6, 2022 at 8:07 AM | -- | Folder | |
| > logs | Jan 7, 2022 at 8:53 AM | Jan 7, 2022 at 8:53 AM | -- | Folder | |
| > objects | Jan 7, 2022 at 3:48 PM | Jan 6, 2022 at 8:07 AM | -- | Folder | |
| > refs | Jan 6, 2022 at 8:07 AM | Jan 6, 2022 at 8:07 AM | -- | Folder | |



Self-Operating Napkin



You do a thing, Git triggers a script



There are 17 available hooks

“Every Git repository has a `.git/hooks` folder with a script for each hook you can bind to. You're free to change or update these scripts as necessary, and Git will execute them when those events occur.” – [Matthew Hudson](https://GitHooks.com)
<https://GitHooks.com>

- `applypatch-msg`
- `pre-applypatch`
- `post-applypatch`
- `pre-commit`
- `prepare-commit-msg`
- `commit-msg`
- `post-commit`
- `pre-rebase`
- `post-checkout`
- `post-merge`
- `pre-receive`
- `update`
- `post-receive`
- `post-update`
- `pre-auto-gc`
- `post-rewrite`
- `pre-push`



**These 3 hooks
trigger before a
commit
happens**



**This one triggers
before the remote
accepts the
changes**



Git comes with sample hooks

```
└─ .git
  └─ hooks
     ├── applypatch-msg.sample
     ├── commit-msg.sample
     ├── fsmonitor-watchman.sample
     ├── post-update.sample
     ├── pre-applypatch.sample
     └─ $ pre-commit.sample
        ├── pre-merge-commit.sample
        ├── pre-push.sample
        ├── pre-rebase.sample
        ├── pre-receive.sample
        ├── prepare-commit-msg.sample
        ├── push-to-checkout.sample
        └── update.sample
```


```
1  #!/bin/sh
2  #
3  # An example hook script to verify what is about to be committed.
4  # Called by "git commit" with no arguments. The hook should
5  # exit with non-zero status after issuing an appropriate message if
6  # it wants to stop the commit.
7  #
8  # To enable this hook, rename this file to "pre-commit".
9
10 if git rev-parse --verify HEAD >/dev/null 2>&1
11 then
12     against=HEAD
13 else
14     # Initial commit: diff against an empty tree object
15     against=$(git hash-object -t tree /dev/null)
16 fi
```



If you can script it, you can automate it.

```
git-venture > .git > hooks > $ commit-msg
1  #!/usr/bin/env bash
2  curl https://icanhazdadjoke.com
3  echo ""
4
```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE



```
Documents/git-venture $ git add .
Documents/git-venture $ git commit -m 'readme update spelling'
Why did the teddy bear say "no" to dessert? Because she was stuffed.
[security da33e2a] readme update spelling
1 file changed, 2 insertions(+), 1 deletion(-)
Documents/git-venture $
```

The built-in scripts are mostly shell (sh) scripts, but you can use ANY scripting language (JavaScript, PHP, Python, Ruby, Perl, Bash, etc);



An ideal solution would look like:

Before I commit,

Something should check my code for any hardcoded:

- Usernames & Passwords
- API Keys
- Security Certificates
- Any other defined patterns

If any of those are detected, **then** throw an error and do not make the commit.



Let's just use git-grep to look for patterns!

```
git-venture > .git > hooks > $ commit-msg
1  #!/usr/bin/env bash
2  curl https://icanhazdadjoke.com
3  echo ""
4  if git grep -E "password = *" | git grep -E "password="* ; then
5  |   echo "NO HARDCODE PASSWORD"
6  |   exit 2
7  | fi
8
9  if git grep -E "[A-Z0-9]{20}"; then
10 |   echo "NO HARDCODE KEYS"
11 |   exit 2
12 | fi
```

```
5  password = exAmplepassWORD
6  password =
7  password=*****
8  More test...
9
10 Key1= AKIAIOSFODNN7HELLOWEKEY
11 Key2 = wJalrXUtnFEMI/K7MDENG/bPxRfiCYHELLOWEKEY
```

```
Documents/git-venture $ git add .
Documents/git-venture $ git commit -m 'readme spacing update'
What did the left eye say to the right eye? Between us, something smells!
README: password=exAmplepassWORD
README: password =
README: password=*****
NO HARDCODE PASSWORD
Documents/git-venture $ █
```



The issues then become...

- We have to manually build and maintain this.
- *False positives!*
- Need to account for “\$API_ENV_VAR” is OK to use in code.
- Account for example passwords (like ‘th1si5@g00Denuff’).
- Tell the difference between long strings and an *access_key_id* or *secret_access_key*.
- Ensure you do not accidentally hard code keys in this scheme in case it gets compromised.
- Provide a way to add more rules and patterns in a sane way.
- Keep track of APIs updates.
- Get your team to adopt your crazy hand rolled scripts.
- Make this more scalable .
- And a bunch of other things you would discover along the way.
- Etc, etc, etc.

Why are you still reading this? Go to the next slide already.

AIN'T
NOBODY
GOT
TIME
FOR THAT



Open Source To The Rescue!



Multiple solutions exist to prevent committing hardcoded secrets

- **AWS-Labs/git-secrets**
- **TruffleHog**
- **ggshield (From Git Guardian)**

Some solutions are built into other security tools

Many more smaller projects



AWS-Labs/git-secrets

git-secrets

🔗 Prevents you from committing passwords and other sensitive information to a git repository.

The following git hooks are installed:

1. `pre-commit` : Used to check if any of the files changed in the commit use prohibited patterns.
2. `commit-msg` : Used to determine if a commit message contains a prohibited patterns.
3. `prepare-commit-msg` : Used to determine if a merge commit will introduce a history that contains a prohibited pattern at any point. Please note that this hook is only invoked for non fast-forward merges.



AWS-Labs/git-secrets

- Free
- Triple checks before a commit is made
- Can be extended
- Relies on dev knowledge of regex and patterns beyond AWS defaults



TruffleHog



TruffleHog

Find leaked credentials.

Precommit Hook

Trufflehog can be used in a precommit hook to prevent credentials from leaking before they ever leave your computer. An example `.pre-commit-config.yaml` is provided (see pre-commit.com for installation).

```
repos:
- repo: local
  hooks:
  - id: trufflehog
    name: TruffleHog
    description: Detect secrets in your data.
    entry: bash -c 'trufflehog git file://. --only-verified --fail'
    # For running trufflehog in docker, use the following entry instead:
    # entry: bash -c 'docker run -v "${pwd}:/workdir" -i --rm trufflesecurity/trufflehog:latest g:
    language: system
    stages: ["commit", "push"]
```



trufflesecurity/trufflehog

- Free
- Checks at the pre-commit level
- Requires the pre-commit framework to be installed as well
- Can run as GitHub Action, catching after secrets make it to remote, assumes GH use
- Reports of high false positives, ymmv



GitGuardian/ggshield

ggshield: protect your secrets with GitGuardian

🔗 The global and local pre-commit hook

To install pre-commit globally (for all current and future repos), run the following command:

```
$ ggshield install --mode global
```

You can also install the hook locally on desired repositories. You just need to go in the repository and execute:

```
$ ggshield install --mode local -t "pre-push"
```

Install ggshield git pre-receive hook



GitGuardian/ggshield

- Requires a GitGuardian account
 - Free for personal and OSS use
- Can be installed at the pre-commit, pre-push, and pre-receive hook levels
- Checks against 350+ known patterns and can be extended, but requires regex skill
- Possible to hit API limits (1000 calls a month on the free plan)



What Does This Look Like In Action?



! config.yml M x ⓘ README.md M \$ 🔗 ↶ ↷ ↻

! config.yml

You, 18 seconds ago | 1 author (You)

```
1 aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxrFi
2 aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxrFi
3
4
```

```
es to be committed:
e "git restore --staged <file>..." to unstage)
new file:   .cache_ggshield
modified:   README
modified:   README.md
modified:   config.yml
new file:   wp-config-sample.php
```

```
enture ~ $git commit -m 'test'
```

```
secrets-engine-version: 2.70.0
```

```
🚫 x 🚫 1 incident has been found in file config.yml
```

```
>>> Incident 1(Secrets detection): Generic High Entropy Secret (Validity: No Checker) (Ignore with SHA: e31b59cd45e66224391d92aef590c7f986296ce46810702d107d6ed784d9f851) (2 occurrences)
```

```
2 | Key2 = AKIAIOSFODNNEXAMPLE
3 | aws_secret_api = wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
1 | aws_secret_access_key = wJalr*****PxrFi
   | _____apikey_____
2 | aws_secret_access_key = wJalr*****PxrFi
   | _____apikey_____
3 |
4 |
```

```
GitGuardian Shield (pre-commit).....
.....Failed
```

Throws an error and does not make a commit!




In Conclusion



- Do NOT hardcode secrets.
- Do NOT commit secrets
- Use automation
- Leverage open source tools to prevent you from pushing secrets

```
1 package main
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 func main() {
9     databaseName := "53CR3TD4T4B453"
10    secretKey := "5UP3R53CR3T"
11    secretPhrase := "Always know where your towel is. - D
12
13    var dbName string
14    var dbPass string
15
16    fmt.Println("Please enter database name:")
17    fmt.Scanf("%s", &dbName)
18
19    fmt.Println("Please enter database password:")
20    fmt.Scanf("%s", &dbPass)
21
22    if dbName == databaseName && dbPass == secretKey {
23        fmt.Println("Welcome to the database!")
24        fmt.Println("Your secret phrase is: ", secretPhrase)
25        os.Exit(0)
26    }
27    fmt.Println("Sorry, wrong database name or password")
28 }
29
```



Hi, I'm Dwayne



Dwayne McDaniel

- I live in Chicago
- I've been a Developer Advocate since 2016
- On Twitter @mcdwayne
- Happy to chat about anything, hit me up
- Besides tech, I love improv, karaoke and going to rock and roll shows!





Stop Committing Your Secrets - Git Hooks To The Rescue

Conf42: DevSecOps 2022

CONF42





Questions?

Let's talk on Twitter @mcdwayne

www.gitguardian.com



@mcdwayne