CAN YOU KEEP A SECRET?

@AARONBASSETT

mongoDB®

AARON BASSETT

SENIOR DEVELOPER ADVOCATE

mongoDB®

```
import pprint
from pymongo import MongoClient

client = MongoClient(
    "C01.5tsil.mongodb.net",
    username="admin", password="hunter2"
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import pprint
from pymongo import MongoClient

client = MongoClient(
    "C01.5tsil.mongodb.net",
    username="admin", password="hunter2"
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import pprint
from pymongo import MongoClient

client = MongoClient(
    "C01.5tsil.mongodb.net",
    username="admin", password="hunter2"
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import pprint
from pymongo import MongoClient

client = MongoClient(
    "C01.5tsil.mongodb.net",
    username="admin", password="hunter2"
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import pprint
from pymongo import MongoClient

client = MongoClient(
    "C01.5tsil.mongodb.net",
    username="admin", password="hunter2"
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import pprint
from pymongo import MongoClient

DB_HOST = "C01.5tsil.mongodb.net"
DB_USERNAME = "admin"
DB_PASSWORD = "hunter2"

client = MongoClient(DB_HOST, username=DB_USERNAME, password=DB_PASSWORD)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import pprint
from pymongo import MongoClient

DB_HOST = "C01.5tsil.mongodb.net"
DB_USERNAME = "admin"
DB_PASSWORD = "hunter2"

client = MongoClient(DB_HOST, username=DB_USERNAME, password=DB_PASSWORD)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
git add .
git commit -m “wip”
git push
```

mongoDB®

"THE MEDIAN TIME TO DISCOVERY WAS 20 SECONDS, WITH TIMES RANGING FROM HALF A SECOND TO OVER 4 MINUTES"

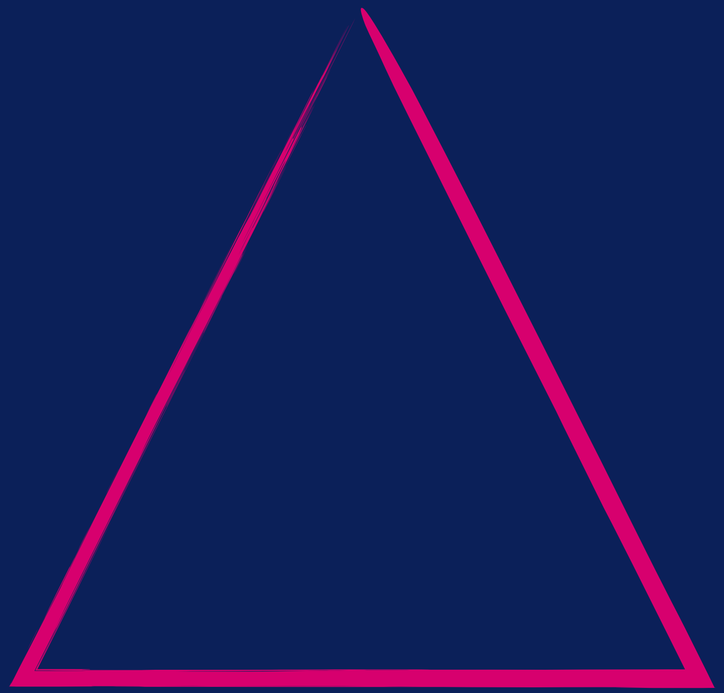
– How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories

mongoDB®

NOT SAFE, BUT VERY EASY

mongoDB®

FUNCTIONALITY



SECURITY

USABILITY

mongoDB®

LOW FRICTION

mongoDB®

EASY TO IMPLEMENT

mongoDB®

12 FACTOR APPS

mongoDB®

I. CODEBASE

II. DEPENDENCIES

III. CONFIG

IV. BACKING SERVICES

V. BUILD, RELEASE, RUN

VI. PROCESSES

VII. PORT BINDING

VIII. CONCURRENCY

IX. DISPOSABILITY

X. DEV/PROD PARITY

XI. LOGS

XII. ADMIN PROCESSES

mongoDB®

III. CONFIG

mongoDB®

"A litmus test for whether an app has all config correctly factored out of the code is whether the codebase could be made open source at any moment, without compromising any credentials."

—The twelve-factor app

mongoDB®

ENVIRONMENT VARIABLES

mongoDB®

```
import os
import pprint
from pymongo import MongoClient

client = MongoClient(
    os.environ["DB_HOST"],
    username=os.environ["DB_USERNAME"],
    password=os.environ["DB_PASSWORD"],
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import os
import pprint
from pymongo import MongoClient

client = MongoClient(
    os.environ["DB_HOST"],
    username=os.environ["DB_USERNAME"],
    password=os.environ["DB_PASSWORD"],
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import os
import pprint
from pymongo import MongoClient

client = MongoClient(
    os.environ.get("DB_HOST"),
    username=os.environ.get("DB_USERNAME"),
    password=os.environ.get("DB_PASSWORD"),
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import os
import pprint
from pymongo import MongoClient

client = MongoClient(
    os.environ.get("DB_HOST"),
    username=os.environ.get("DB_USERNAME"),
    password=os.environ.get("DB_PASSWORD"),
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
import os
import pprint
from pymongo import MongoClient

client = MongoClient(
    os.getenv("DB_HOST"),
    username=os.getenv("DB_USERNAME"),
    password=os.getenv("DB_PASSWORD"),
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```python
import os
import pprint
from pymongo import MongoClient

client = MongoClient(
    os.getenv("DB_HOST"),
    username=os.getenv("DB_USERNAME"),
    password=os.getenv("DB_PASSWORD"),
)
db = client.geo_example

query = {"loc": {"$within": {"$center": [[0, 0], 6]}}}
for doc in db.places.find(query).sort("_id"):
    pprint.pprint(doc)
```

mongoDB®

```
def getenv(key, default=None):
    """Get an environment variable, return None if it doesn't exist.
    The optional second argument can specify an alternate default.
    key, default and the result are str."""
    return environ.get(key, default)
```

```
def getenv(key, default=None):
    """Get an environment variable, return None if it doesn't exist.
    The optional second argument can specify an alternate default.
    key, default and the result are str."""
    return environ.get(key, default)
```

CREATING ENVIRONMENT VARIABLES

mongoDB®

```
# This file must be used with "source bin/activate" *from bash*
# you cannot run it directly

deactivate () {
    ...

    # Unset variables
    unset NEXMO_KEY
    unset NEXMO_SECRET
    unset MY_NUMBER
}

...

export NEXMO_KEY="a925db1ar392"
export NEXMO_SECRET="01nd637fn29oe31mc721"
export MY_NUMBER="447700900981"
```

mongoDB®

```
# This file must be used with "source bin/activate" *from bash*
# you cannot run it directly

deactivate () {
     ...

    # Unset variables
    unset NEXMO_KEY
    unset NEXMO_SECRET
    unset MY_NUMBER
}

 ...

export NEXMO_KEY="a925db1ar392"
export NEXMO_SECRET="01nd637fn29oe31mc721"
export MY_NUMBER="447700900981"
```

```
# This file must be used with "source bin/activate" *from bash*
# you cannot run it directly

deactivate () {
    ...

    # Unset variables
    unset NEXMO_KEY
    unset NEXMO_SECRET
    unset MY_NUMBER
}

...

export NEXMO_KEY="a925db1ar392"
export NEXMO_SECRET="01nd637fn29oe31mc721"
export MY_NUMBER="447700900981"
```

mongoDB®

"direnv is an extension for your shell. It augments existing shells with a new feature that can load and unload environment variables depending on the current directory."

– direnv.net

mongoDB®

```
$ echo export DB_PASSWORD=hunter2 > .envrc
```

mongoDB®

```
$ echo export DB_PASSWORD=hunter2 > .envrc
.envrc is not allowed
```

```
$ echo export DB_PASSWORD=hunter2 > .envrc
.envrc is not allowed
$ direnv allow .
```

mongoDB®

```
$ echo export DB_PASSWORD=hunter2 > .envrc
.envrc is not allowed
$ direnv allow .
direnv: reloading
direnv: loading .envrc
direnv export: +DB_PASSWORD
```

mongoDB®

```
$ echo export DB_PASSWORD=hunter2 > .envrc
.envrc is not allowed
$ direnv allow .
direnv: reloading
direnv: loading .envrc
direnv export: +DB_PASSWORD
$ cd ..
```

mongoDB®

```
$ echo export DB_PASSWORD=hunter2 > .envrc
.envrc is not allowed
$ direnv allow .
direnv: reloading
direnv: loading .envrc
direnv export: +DB_PASSWORD
$ cd ..
direnv: unloading
```

mongoDB®

IGNORE .ENVRC

```
echo .envrc > ~/.gitignore
git config --global core.excludesfile ~/.gitignore
```

mongoDB®

EXAMPLE .ENVRC

```
grep -ohr "^export .*=" .envrc > .envrc.example
```

mongoDB®

SHARING SECRET FILES

mongoDB®

ENCRYPTION AT REST

mongoDB®

CLIENT-SIDE FIELD LEVEL ENCRYPTION

mongoDB®

USE A KMS

mongoDB®

```
openssl rand -base64 32 > mongodb-keyfile
chmod 600 mongodb-keyfile
mongod --enableEncryption --encryptionKeyFile  mongodb-keyfile
```

git-secret

GIT-SECRET.IO

mongoDB®

"git-secret encrypts files and stores them inside the git repository, so you will have all the changes for every commit."

—git-secret.io

mongoDB®

SAFE & EASY

mongoDB®

SAFE & EASY-ISH

mongoDB®

PGP

mongoDB®

```
$ git secret init
git-secret: init created: '/myproject/.gitsecret/'
```

mongoDB®

```
$ git secret tell me@aaronbassett.com
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2021-09-30
gpg: keybox '/myproject/.gitsecret/keys/pubring.kbx' created
gpg: /myproject/.gitsecret/keys/trustdb.gpg: trustdb created
git-secret: done. me@aaronbassett.com added as user(s) who know the secret.
```

```
$ git secret add mongodb-keyfile
git-secret: these files are not in .gitignore: mongodb-keyfile
git-secret: auto adding them to .gitignore
git-secret: 1 item(s) added.
```

```
$ cat .gitignore
.gitsecret/keys/random_seed
!*.secret
mongodb-keyfile
```

mongoDB®

```
$ ls
.git
.gitignore
.gitsecret
mongodb-keyfile
```

mongoDB®

```
$ git secret hide
git-secret: done. 1 of 1 files are hidden.
```

mongoDB®

```
$ ls
.git
.gitignore
.gitsecret
mongodb-keyfile
mongodb-keyfile.secret
```

mongoDB®

```
$ git secret reveal
File '/myproject/mongodb-keyfile' exists. Overwrite? (y/N) y
git-secret: done. 1 of 1 files are revealed.
```

mongoDB®

```
$ git secret whoknows
me@aaronbassett.com

$ git secret list
mongodb-keyfile
```

mongoDB®

```
$ git secret killperson me@aaronbassett.com
git-secret: removed keys.
git-secret: now [me@aaronbassett.com] do not have an access to the repository.
git-secret: make sure to hide the existing secrets again.

$ git secret reveal
git-secret: abort: no public keys for users found. run 'git secret tell email@address.
```

GIT SECRETS

"git-secrets scans commits, commit messages, and --no-ff merges to prevent adding secrets into your git repositories. If a commit, commit message, or any commit in a --no-ff merge history matches one of your configured prohibited regular expression patterns, then the commit is rejected."

– awslabs/git-secrets

mongoDB®

```
$ git secrets --register-aws --global
OK

$ git secrets --install ~/.git-templates/git-secrets
√ Installed commit-msg hook to /Users/aaronbassett/.git-templates/git-
secrets/hooks/commit-msg
√ Installed pre-commit hook to /Users/aaronbassett/.git-templates/git-
secrets/hooks/pre-commit
√ Installed prepare-commit-msg hook to /Users/aaronbassett/.git-
templates/git-secrets/hooks/prepare-commit-msg

$ git config --global init.templateDir ~/.git-templates/git-secrets
```

mongoDB®

```
$ git secrets --register-aws --global
OK

$ git secrets --install ~/.git-templates/git-secrets
✓ Installed commit-msg hook to /Users/aaronbassett/.git-templates/git-secrets/hooks/commit-msg
✓ Installed pre-commit hook to /Users/aaronbassett/.git-templates/git-secrets/hooks/pre-commit
✓ Installed prepare-commit-msg hook to /Users/aaronbassett/.git-templates/git-secrets/hooks/prepare-commit-msg

$ git config --global init.templateDir ~/.git-templates/git-secrets
```

mongoDB®

```
$ git secrets --register-aws --global
OK

$ git secrets --install ~/.git-templates/git-secrets
✓ Installed commit-msg hook to /Users/aaronbassett/.git-templates/git-secrets/hooks/commit-msg
✓ Installed pre-commit hook to /Users/aaronbassett/.git-templates/git-secrets/hooks/pre-commit
✓ Installed prepare-commit-msg hook to /Users/aaronbassett/.git-templates/git-secrets/hooks/prepare-commit-msg

$ git config --global init.templateDir ~/.git-templates/git-secrets
```

mongoDB®

PROVIDERS

mongoDB®

```
^[5KL][1-9A-HJ-NP-Za-km-z]{50,51}$
(xox[p|b|o|a]-[0-9]{12}-[0-9]{12}-[0-9]{12}-[a-z0-9]{32})
https://hooks.slack.com/services/T[a-zA-Z0-9_]{8}/B[a-zA-Z0-9_]{8}/[a-zA-Z0-9_]{24}
EAACEdEose0cBA[0-9A-Za-z]+
[t|T][w|W][i|I][t|T][t|T][e|E][r|R].*[1-9][0-9]+-[0-9a-zA-Z]{40}
[t|T][w|W][i|I][t|T][t|T][e|E][r|R].*['|\"][0-9a-zA-Z]{35,44}['|\"]
AIza[0-9A-Za-z\\-_]{35}
[0-9]+-[0-9A-Za-z_]{32}\\.apps\\.googleusercontent\\.com
ya29\\.[0-9A-Za-z\\-_]+
[h|H][e|E][r|R][o|O][k|K][u|U].*[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}
(-----(BEGIN|END) PRIVATE KEY-----)
(-----(BEGIN|END) RSA PRIVATE KEY-----)
```

```
$ git secrets --add-provider -- cat /secret/file/patterns
```

mongoDB®

(-----(BEGIN|END) RSA PRIVATE KEY-----)

mongoDB®

```
ya29\\.[0-9A-Za-z\\-_]+
```

mongoDB®

EAACEdEose0cBA[0-9A-Za-z]+

mongoDB®

```
$ cd /secret/file/patterns
$ ls
crypto
keys
vendors
```

mongoDB®

```
# Slack
(xox[p|b|o|a]-[0-9]{12}-[0-9]{12}-[0-9]{12}-[a-z0-9]{32})
https://hooks.slack.com/services/T[a-zA-Z0-9_]{8}/B[a-zA-Z0-9_]{8}/[a-zA-Z0-9_]{24}

# Facebook
EAACEdEose0cBA[0-9A-Za-z]+

# Twitter
[t|T][w|W][i|I][t|T][t|T][e|E][r|R].*[1-9][0-9]+-[0-9a-zA-Z]{40}
[t|T][w|W][i|I][t|T][t|T][e|E][r|R].*['|\"][0-9a-zA-Z]{35,44}['|\"]

# Google
AIza[0-9A-Za-z\\-_]{35}
[0-9]+-[0-9A-Za-z_]{32}\\.apps\\.googleusercontent\\.com
ya29\\.[0-9A-Za-z\\-_]+

# Heroku
[h|H][e|E][r|R][o|O][k|K][u|U].*[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}
```

mongoDB®

```
git secrets --add-provider -- egrep -rhv "(^#|^$)" /secret/file/patterns
```

mongoDB®

```
^[5KL][1-9A-HJ-NP-Za-km-z]{50,51}$
(xox[p|b|o|a]-[0-9]{12}-[0-9]{12}-[0-9]{12}-[a-z0-9]{32})
https://hooks.slack.com/services/T[a-zA-Z0-9_]{8}/B[a-zA-Z0-9_]{8}/[a-zA-Z0-9_]{24}
EAACEdEose0cBA[0-9A-Za-z]+
[t|T][w|W][i|I][t|T][t|T][e|E][r|R].*[1-9][0-9]+-[0-9a-zA-Z]{40}
[t|T][w|W][i|I][t|T][t|T][e|E][r|R].*['|\"][0-9a-zA-Z]{35,44}['|\"]
AIza[0-9A-Za-z\\-_]{35}
[0-9]+-[0-9A-Za-z_]{32}\\.apps\\.googleusercontent\\.com
ya29\\.[0-9A-Za-z\\-_]+
[h|H][e|E][r|R][o|O][k|K][u|U].*[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}
(-----(BEGIN|END) PRIVATE KEY-----)
(-----(BEGIN|END) RSA PRIVATE KEY-----)
```

```
$ git add ‘private.key’
$ git commit -m "Adding some files I shouldn't"
private.key:1:-----BEGIN PRIVATE KEY-----

[ERROR] Matched one or more prohibited patterns

Possible mitigations:
- Mark false positives as allowed using: git config --add secrets.allowed ...
- Mark false positives as allowed by adding regular expressions to .gitallowed at repository's root directory
- List your configured patterns: git config --get-all secrets.patterns
- List your configured allowed patterns: git config --get-all secrets.allowed
- List your configured allowed patterns in .gitallowed at repository's root directory
- Use --no-verify if this is a one-time false positive
```

GITLEAKS

mongoDB®

"AUDIT GIT REPOS FOR SECRETS. GITLEAKS PROVIDES A WAY FOR YOU TO FIND UNENCRYPTED SECRETS AND OTHER UNWANTED DATA TYPES IN GIT SOURCE CODE REPOSITORIES"

— Zri cEthEZaV/9I tLEaKS

mongoDB®

I. A GIT REPO

II. GITHUB USER

III. GITHUB ORGANIZATION

IV. GITHUB PR

V. GITLAB USER

VI. GITLAB GROUP

mongoDB®

```
{
    "line": "-----BEGIN PRIVATE KEY-----",
    "commit": "37f19780583f12fd2fb687f2d7d3840880e79c76",
    "offender": "-----BEGIN PRIVATE KEY-----",
    "rule": "PKCS8",
    "info": "-----BEGIN PRIVATE KEY----- regex match",
    "commitMsg": "wip backup ",
    "author": "Joe Bloggs",
    "email": "jbloggs@example.com",
    "file": "app/private.key",
    "repo": "my-awesome-app",
    "date": "2019-09-14T12:26:10+08:00",
    "tags": "key, PKCS8",
    "severity": ""
  }
```


mongoDB®

1. KEEP SECRETS AND CODE SEPARATE

mongoDB®

2. IF YOU NEED TO SHARE SECRETS ENCRYPT THEM FIRST

PGP IS A PITA, USE TOOLS TO MAKE IT EASIER

mongoDB®

3. AUTOMATE, AUTOMATE, AUTOMATE

mongoDB®

4. LATE IS BETTER THAN NEVER

mongoDB®

@AARONBASSETT
NOTI.ST/AARONBASSETT

mongoDB®