
THE SECRETS IN YOUR
GIT REPOSITORY

06.10.2021

06.10.2021

6000 internal Twitch.tv git repositories
were leaked to the public

Inside the repositories researchers found:

Inside the repositories researchers found:

- 194 AWS keys

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys
- 68 Google API keys

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys
- 68 Google API keys
- 14 Github OAuth keys

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys
- 68 Google API keys
- 14 Github OAuth keys
- 4 Stripe keys

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys
- 68 Google API keys
- 14 Github OAuth keys
- 4 Stripe keys
- 1000s of passwords

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys
- 68 Google API keys
- 14 Github OAuth keys
- 4 Stripe keys
- 1000s of passwords
- 100s of database connection strings

Inside the repositories researchers found:

- 194 AWS keys
- 69 Twilio keys
- 68 Google API keys
- 14 Github OAuth keys
- 4 Stripe keys
- 1000s of passwords
- 100s of database connection strings
- 100s of private keys

WHAT IF ATTACKERS HAD ACCESS TO YOUR INTERNAL REPOSITORIES?

WHAT IF ATTACKERS HAD ACCESS TO YOUR INTERNAL REPOSITORIES?

... would you be worried?

TOPICS

TOPICS

- Why are there credentials in git repositories? 🤔

TOPICS

- Why are there credentials in git repositories? 🤔
- Tools to find credentials 🔍

TOPICS

- Why are there credentials in git repositories? 🤔
- Tools to find credentials 🔍
- Mining public git repositories for AWS keys ⛏️

TOPICS

- Why are there credentials in git repositories? 🤔
- Tools to find credentials 🔍
- Mining public git repositories for AWS keys ⛏️
- Where else can credentials be found? 🤷‍♀️

TOPICS

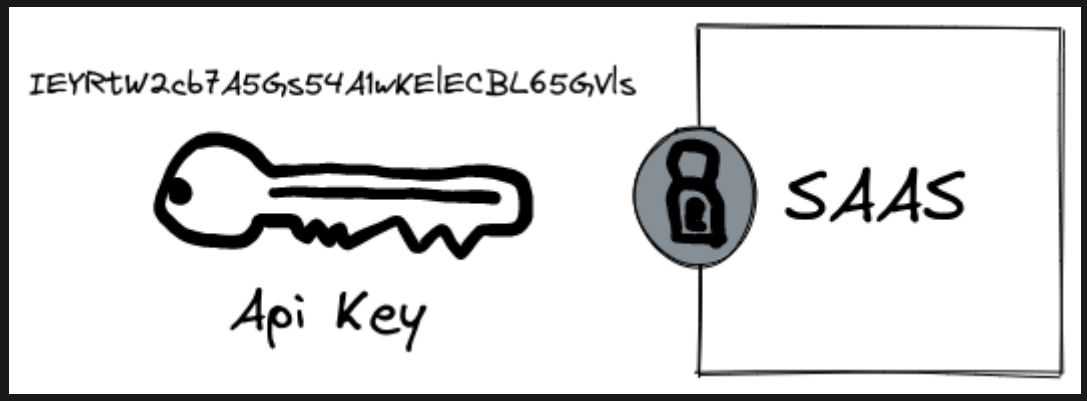
- Why are there credentials in git repositories? 🤔
- Tools to find credentials 🔍
- Mining public git repositories for AWS keys ⛏️
- Where else can credentials be found? 🤷
- API keys and cloud security ☁️

TOPICS

- Why are there credentials in git repositories? 🤔
- Tools to find credentials 🔍
- Mining public git repositories for AWS keys ⛏️
- Where else can credentials be found? 🤷‍♀️
- API keys and cloud security ☁️
- How can I protect myself? 🛡️

WHY ARE THERE CREDENTIALS IN GIT REPOSITORIES? 🤔

HOW SECRETS ARE USED



Secrets are used to authenticate software against external services.

Venue and Venue Place Crud added

main

Showing 10 changed files with 1,325 additions and 108 deletions.

16 Services/S3.js

```diff
@@ -1,11 +1,9 @@
- import AWS from "aws-sdk";
-
- export const S3 = {
-   accessKeyId: "",
-   secretAccessKey: "",
+ console.log("Process ::", process.env.NEXT_AWS_ACCESS_KEY);
+ const S3 = {
+   accessKeyId: process.env.NEXT_AWS_ACCESS_KEY || "AKIA[redacted]",
+   secretAccessKey:
+     process.env.NEXT_AWS_SECRET_KEY ||
+     "[redacted]",
  };
```

Secrets are committed by accident

THE 4 MATURITY LEVELS OF SECRET REMOVAL

LEVEL 0

Credentials are not deleted

LEVEL 1

A commit deletes the secret from the code. It still persists in the git history.

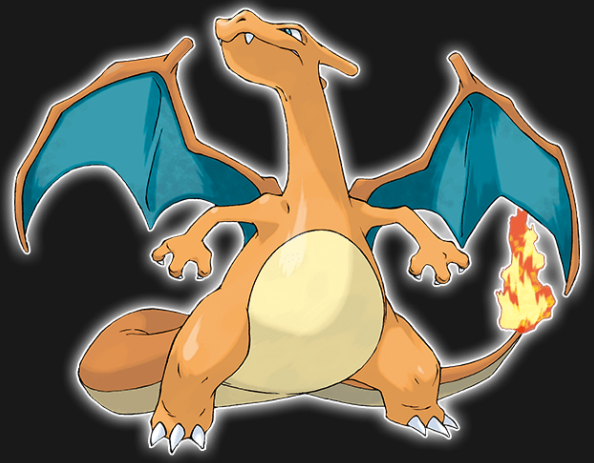
6 Services/S3.js

```diff
@@ -1,9 +1,7 @@
console.log("Process ::", process.env.NEXT_AWS_ACCESS_KEY);
const S3 = {
-   accessKeyId: process.env.NEXT_AWS_ACCESS_KEY || "AKIA[redacted]",
-   secretAccessKey:
-     process.env.NEXT_AWS_SECRET_KEY ||
-     "[redacted]",
+   accessKeyId: process.env.NEXT_AWS_ACCESS_KEY,
+   secretAccessKey: process.env.NEXT_AWS_SECRET_KEY,
};

export default S3;
```

Still part of the history

LEVEL 2

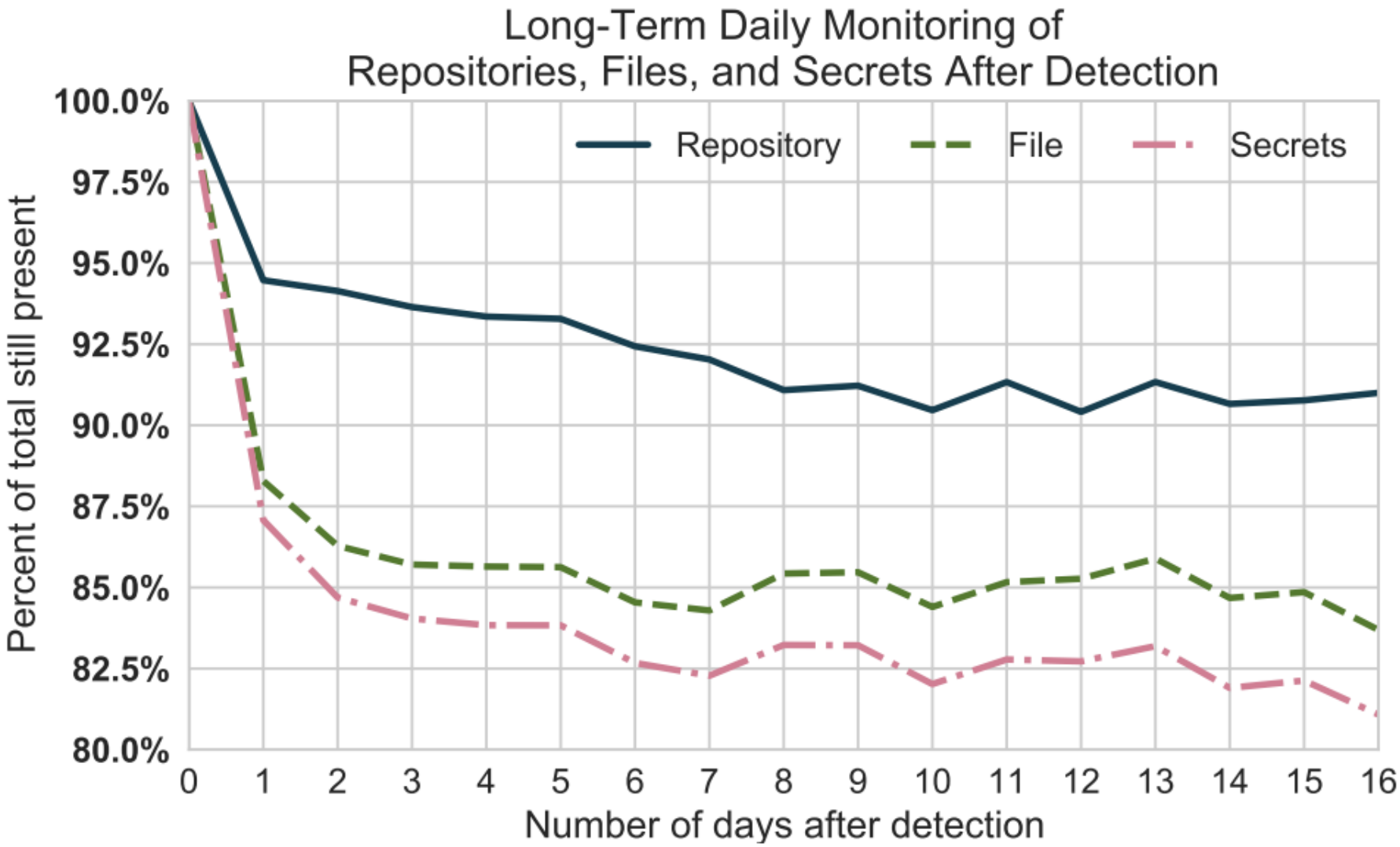


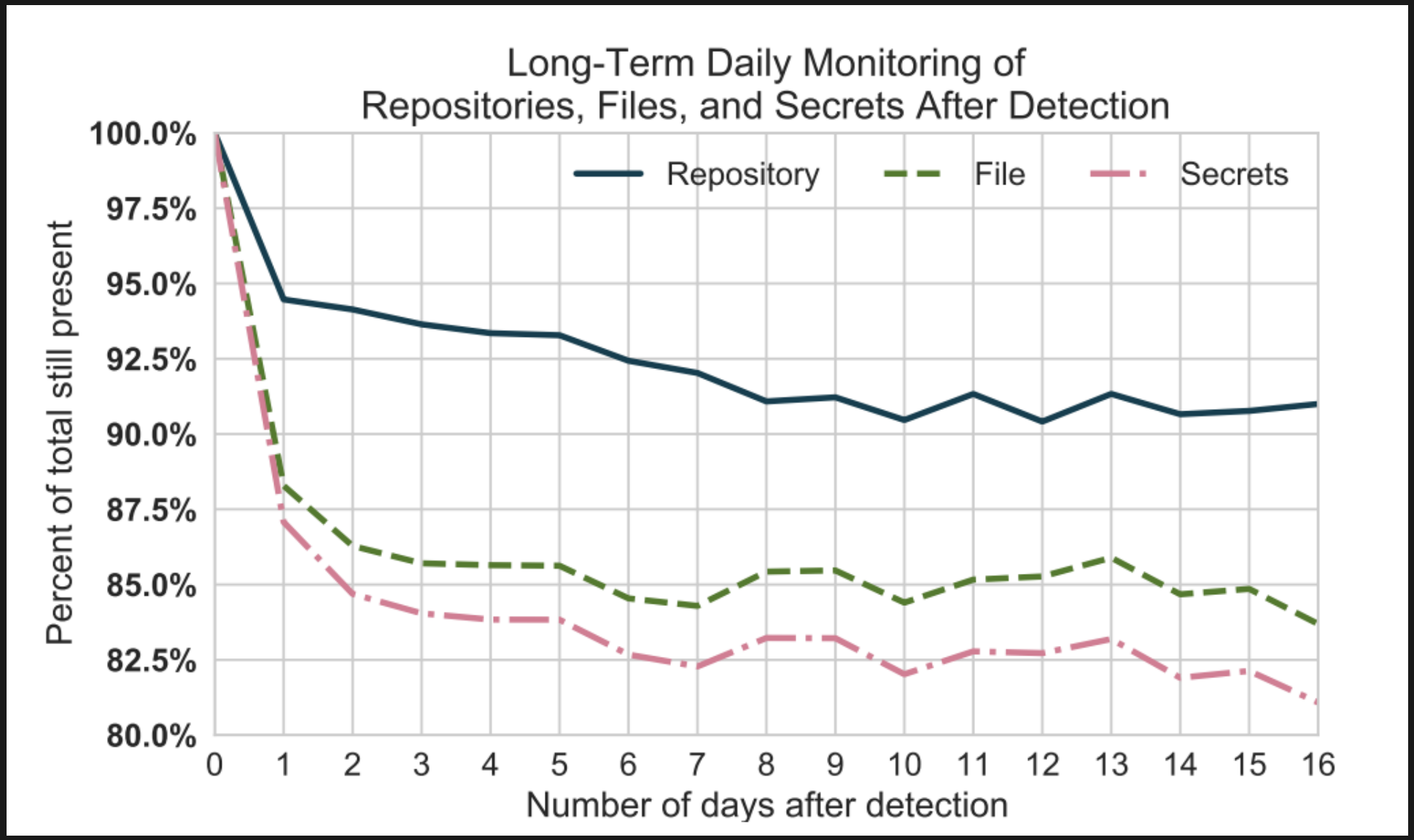
Git history is rewritten. Still vulnerable to monitoring and advanced repository analysis techniques.

LEVEL 3

Secret is rotated. Can be painful and time consuming.

DID YOU KNOW...


Long-Term Daily Monitoring of
Repositories, Files, and Secrets After Detection
Repository
File
Secrets
Percent of total still present
100.0%
97.5%
95.0%
92.5%
90.0%
87.5%
85.0%
82.5%
80.0%
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Number of days after detection



Secrets not deleted after 24 hours tend to stay forever

GIT REPOSITORIES ARE EVERYWHERE

GIT REPOSITORIES ARE EVERYWHERE

- Configuration issue of the git server could make all repositories public

GIT REPOSITORIES ARE EVERYWHERE

- Configuration issue of the git server could make all repositories public
- Local copies are shared

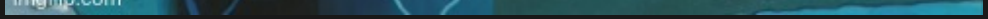
GIT REPOSITORIES ARE EVERYWHERE

- Configuration issue of the git server could make all repositories public
- Local copies are shared
- Build server misconfiguration pushes git repositories to artifact stores

GIT REPOSITORIES ARE EVERYWHERE

- Configuration issue of the git server could make all repositories public
- Local copies are shared
- Build server misconfiguration pushes git repositories to artifact stores
- Repositories are deployed on a web server


.GIT
.GIT
.GIT
.GIT
.GIT
.GIT
imgflip.com



TOOLS TO FIND CREDENTIALS 🔎

HOW DO THEY WORK?

HOW DO THEY WORK?

HEURISTICS

HOW DO THEY WORK?

HEURISTICS

- Pattern matching (regex)

HOW DO THEY WORK?

HEURISTICS

- Pattern matching (regex)
- Entropy

HOW DO THEY WORK?

HEURISTICS

- Pattern matching (regex)
- Entropy
- Context information

TRUFFLEHOG

- New Version 3 was released recently
- Scans the history of a repository for secrets
- Over 700 credential detectors
- Support for active verification

CREDENTIALS DETECTORS

```
idPat  = regexp.MustCompile(`\b((?:AKIA|ABIA|ACCA|ASIA)[0-9A-Z
keyPat = regexp.MustCompile(`\b([A-Za-z0-9+/]{40})\b`)
```

ACTIVE VERIFICATION

```
1  // REQUEST VALUES.
2  method := "GET"
3  service := "sts"
4  host := "sts.amazonaws.com"
5  region := "us-east-1"
6  endpoint := "https://sts.amazonaws.com"
7  datestamp := time.Now().UTC().Format("20060102")
8  amzDate := time.Now().UTC().Format("20060102T150405Z0700")
9
10 req, err := http.NewRequestWithContext(ctx, method, endpoir
11 if err != nil {
12   continue
13 }
14
15 // TASK 1: CREATE A CANONICAL REQUEST
```

ACTIVE VERIFICATION

```
16 // http://docs.aws.amazon.com/general/latest/gr/sigv4-creat
17 canonicalURI := "/"
18 canonicalHeaders := "host:" + host + "\n"

19 signedHeaders := "host"
20 algorithm := "AWS4-HMAC-SHA256"
21 credentialScope := fmt.Sprintf("%s/%s/%s/aws4_request", dat
22
23 params := req.URL.Query()
24 params.Add("Action", "GetCallerIdentity")
25 params.Add("Version", "2011-06-15")
26 params.Add("X-Amz-Algorithm", algorithm)
27 params.Add("X-Amz-Credential", resIdMatch+"/"+credentialSco
28 params.Add("X-Amz-Date", amzDate)
29 params.Add("X-Amz-Expires", "30")
```

GITLEAKS

- Alternative to trufflehog
- Similar feature set
- Many knobs and buttons
- Does not offer active verification

GIT HOUND

- Built for bug hunters / Audits
- Intensive repository digging
- Results need to be reviewed manually

GitGuardian

- Commercial SaaS provider
- Easy integration in build pipelines
- Additional Features like Alerting and Dashboards

MINING PUBLIC GIT REPOSITORIES FOR AWS KEYS

Project Report

GOALS OF PROJECT

GOALS OF PROJECT

- Verify theory that many Developers don't know how to delete accidental secret commits.

GOALS OF PROJECT

- Verify theory that many Developers don't know how to delete accidental secret commits.
- Verify theory that there are many active credentials present on public git repositories.

GOALS OF PROJECT

- Verify theory that many Developers don't know how to delete accidental secret commits.
- Verify theory that there are many active credentials present on public git repositories.
- Search for AWS keys.

RULES OF ENGAGEMENT

RULES OF ENGAGEMENT

- No destructive operations with obtained credentials

RULES OF ENGAGEMENT

- No destructive operations with obtained credentials
- No accessing / exfiltration of any data

RULES OF ENGAGEMENT

- No destructive operations with obtained credentials
- No accessing / exfiltration of any data
- Use credentials only to verify their permission level

RULES OF ENGAGEMENT

- No destructive operations with obtained credentials
- No accessing / exfiltration of any data
- Use credentials only to verify their permission level
- Notify repository owners afterwards


HERE'S THE PLAN!
THE SCHEME: LOUIE'S ELEVEN
#1
#2
#3
#6
#7
#8
#10
#11
DUCKBURG DAILIES

STEP 1: QUERIES

Query Strings

```
[
    'remove aws key',
    'delete aws key',
    'remove aws credentials',
    'remove aws secret',
    'remove s3 key',
    'delete s3 key',
    'remove s3 credentials',
    'remove s3 secret'
]
```

Run queries against Github Search API

STEP 2: CLEAN UP REPOSITORY LIST

Repositories

https://github.com/EthanJY-CS/web
https://github.com/vivmehra/graph
https://github.com/DiversityDatabase
https://github.com/thanhwoe/Future
https://github.com/cholzsupermind/IDS
https://github.com/khushnoodasif/aws
https://github.com/GaneshHub/CA
https://github.com/desenvolvimento/v1
https://github.com/vladtwork/garage

First 200 results of each query. Duplicates removed

STEP 3: MINE CREDENTIALS

Credential List


```
{
  "SourceID": 0,
  "SourceType": 7,
  "SourceName": "trufflehog - github",
  "DetectorType": 2,
  "DetectorName": "AWS",
  "Verified": false,
  "Raw": "QUtJQTNXNVhDT1lPVUNGNlFEWFg=",
  "Redacted": "AKIA3W5XCOYOUCF6QDXX",
  "ExtraData": null,
  "StructuredData": null
}
```


Repositories >50MB were skipped.

STEP 4: CLEAN UP CREDENTIAL LIST

AWS Keys


```
{
  "SourceID": 0,
  "SourceType": 7,
  "SourceName": "trufflehog - github",
  "DetectorType": 2,
  "DetectorName": "AWS",
  "Verified": true,
  "Raw": "[redacted]",
  "Redacted": "AKIA[redacted]",
  "ExtraData": null,
  "StructuredData": null
}
```


Unique, verified AWS keys.

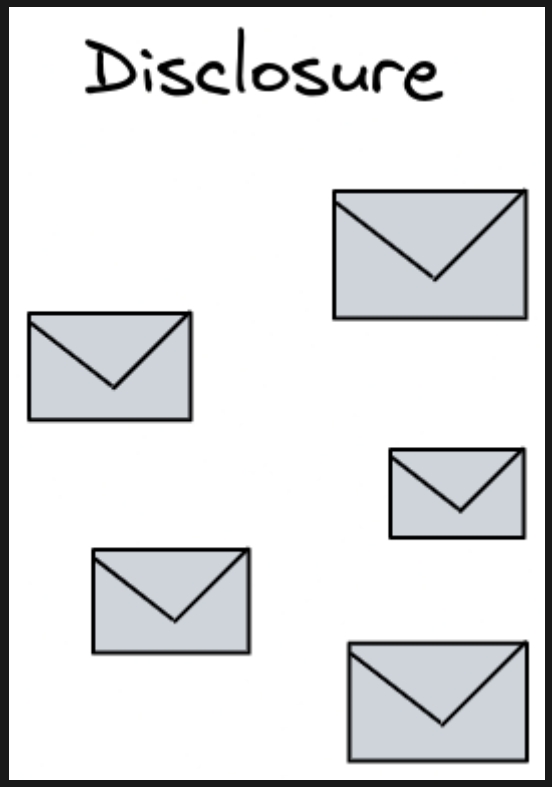
STEP 5: VERIFY KEY PRIVILEGES

Key Privileges


```
{
  "KeyId": "AKIA[redacted]",
  "s3List": false,
  "ec2List": false,
  "dynamoList": true,
  "iamList": true
}
```


Only test read access to resources.

STEP 6: NOTIFY REPOSITORY OWNERS



Send notification emails with instructions how to clean up.


ITS GO TIME!
vitalfog

8 GITHUB QUERIES

1480 UNIQUE REPOSITORIES

61 UNIQUE, VERIFIED AWS KEYS

BYCATCH

SonarCloud
PrivateKeys
SendGrid
Aplitude
Redis
Discord
GoogleCloud
Algolia
Alibaba
Mailchimp
Flickr
Facebook
Circle

OF THE 61 AWS KEYS

OF THE 61 AWS KEYS

- 13 Keys with S3 read access

OF THE 61 AWS KEYS

- 13 Keys with S3 read access
- 26 Keys with DynamoDB read access

OF THE 61 AWS KEYS

- 13 Keys with S3 read access
- 26 Keys with DynamoDB read access
- 25 Keys with EC2 read access

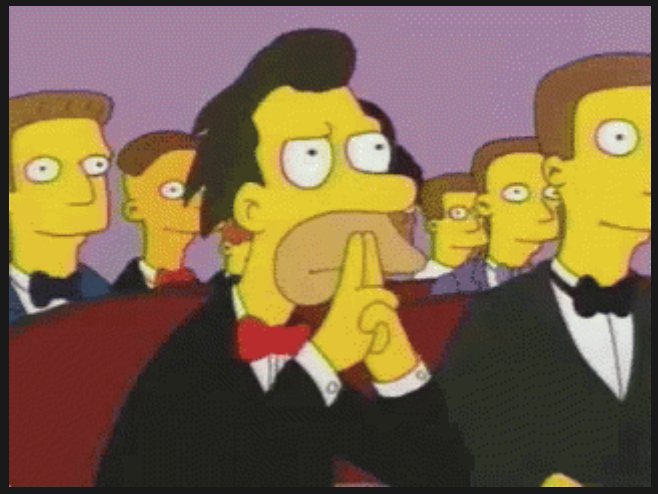
OF THE 61 AWS KEYS

- 13 Keys with S3 read access
- 26 Keys with DynamoDB read access
- 25 Keys with EC2 read access
- 23 Keys with IAM read access

DISCLOSURE RESPONSE

DISCLOSURE RESPONSE

A MORE TARGETED APPROACH



A MORE TARGETED APPROACH

A MORE TARGETED APPROACH

1. Use OSINT to find domains and subdomains of organizations

A MORE TARGETED APPROACH

1. Use OSINT to find domains and subdomains of organizations
2. Use github search to find repositories which contain the org domains

A MORE TARGETED APPROACH

1. Use OSINT to find domains and subdomains of organizations
2. Use github search to find repositories which contain the org domains
3. Filter out usages of public APIs

A MORE TARGETED APPROACH

1. Use OSINT to find domains and subdomains of organizations
2. Use github search to find repositories which contain the org domains
3. Filter out usages of public APIs
4. If you find new hostnames, domains and subdomains, repeat

A MORE TARGETED APPROACH

1. Use OSINT to find domains and subdomains of organizations
2. Use github search to find repositories which contain the org domains
3. Filter out usages of public APIs
4. If you find new hostnames, domains and subdomains, repeat
5. Scan all repositories for secrets

A MORE TARGETED APPROACH

1. Use OSINT to find domains and subdomains of organizations
2. Use github search to find repositories which contain the org domains
3. Filter out usages of public APIs
4. If you find new hostnames, domains and subdomains, repeat
5. Scan all repositories for secrets
6. Don't limit search on github

WHERE ELSE CAN CREDENTIALS BE FOUND? 🤷‍♀️

OTHER PLACES TO SEARCH

OTHER PLACES TO SEARCH

- Gitlab

OTHER PLACES TO SEARCH

- Gitlab
- Github Gists

OTHER PLACES TO SEARCH

- Gitlab
- Github Gists
- Publicly exposed git repositories

OTHER PLACES TO SEARCH

- Gitlab
- Github Gists
- Publicly exposed git repositories
- Artifact registries

OTHER PLACES TO SEARCH

- Gitlab
- Github Gists
- Publicly exposed git repositories
- Artifact registries
- Docker images

OTHER PLACES TO SEARCH

- Gitlab
- Github Gists
- Publicly exposed git repositories
- Artifact registries
- Docker images
- Firmware from hardware

OTHER PLACES TO SEARCH

- Gitlab
- Github Gists
- Publicly exposed git repositories
- Artifact registries
- Docker images
- Firmware from hardware
- Hardcoded in Frontend Applications (Web and mobile)

API KEYS AND CLOUD SECURITY

CLOUD HACKING

Very big topic. Different from traditional attacks.
Typically does not use malware and exploits

TYPICAL ATTACK

TYPICAL ATTACK

1. Initial access: Leaked credential or OAuth phishing

TYPICAL ATTACK

1. Initial access: Leaked credential or OAuth phishing
2. Enumerate permissions

TYPICAL ATTACK

1. Initial access: Leaked credential or OAuth phishing
2. Enumerate permissions
3. Escalate permissions by finding / creating new credentials

TYPICAL ATTACK

1. Initial access: Leaked credential or OAuth phishing
2. Enumerate permissions
3. Escalate permissions by finding / creating new credentials
4. Repeat

HOW CAN I PROTECT MYSELF?

MITIGATION

MITIGATION

- Onion approach works best

MITIGATION

- Onion approach works best
- Developer education

MITIGATION

- Onion approach works best
- Developer education
- Code reviews

MITIGATION

- Onion approach works best
- Developer education
- Code reviews
- Use tooling to detect credentials in build pipeline

MITIGATION

- Onion approach works best
- Developer education
- Code reviews
- Use tooling to detect credentials in build pipeline
- Try to keep the false positives low!

MITIGATION

- Onion approach works best
- Developer education
- Code reviews
- Use tooling to detect credentials in build pipeline
- Try to keep the false positives low!
- Check for secrets in code before open sourcing or sharing with third parties

MITIGATION

MITIGATION

- Follow least privilege principle for service accounts

MITIGATION

- Follow least privilege principle for service accounts
- Don't reuse secrets

MITIGATION

- Follow least privilege principle for service accounts
- Don't reuse secrets
- Store production credentials in credential stores like vault.

MITIGATION

- Follow least privilege principle for service accounts
- Don't reuse secrets
- Store production credentials in credential stores like vault.
- Use CloudTrail / Activity log to monitor activity of keys

MITIGATION

- Follow least privilege principle for service accounts
- Don't reuse secrets
- Store production credentials in credential stores like vault.
- Use CloudTrail / Activity log to monitor activity of keys
- Delete keys when they are no longer required

BE LIKE GANDALF

SOURCES

- Twitch Leaks research
- How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories
- Shannon entropy
- TruffleHog
- GitLeaks
- Git Hound
- Credential mining Bug Bounties
- Github Dorking
- SantaHog

SOURCES (CONTINUED)

- Cloud Hacking - Malware not needed
- AWS Cloud trail

IMAGE SOURCES

- Title page background
- Pokemon

PRESENTATION IS PUBLISHED ON GITHUB

https://github.com/nixrod/credential-harvesting-presentation