HITB SECCONF 2023 Phuket

# Big Match:

## How I Learned to Stop Reversing and Love the Strings

Paolo MONTESEL / babush
Myself, Inc

# Abstract

We’ve all been there: after a month of reversing, you realize you are looking at **open-source code.**

Why?

Because you didn’t copy-paste the correct string into Google.

So we asked ourselves: **“can’t we not just grep all strings from GitHub and stop this nonsense?”**

# About myself

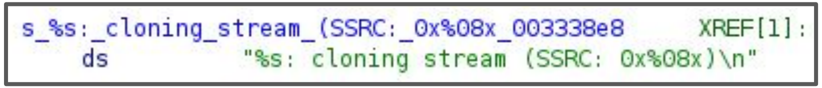
- Self-employed security and whatnot guy
- Reversing ∩ data-science ∩ ML
- Interested in data-driven stuff
- Previous research: MikroTik, Naver LINE, Bison/Flex parsers, other
- CTF player for mhackeroni
  - Just won Hack-A-Sat :D

# About rev.ng

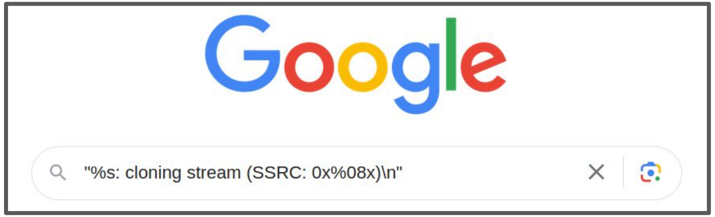
- https://rev.ng/
- Building an LLVM-based decompiler
- Binary analysis, reverse engineering
- C++/LLVM consulting
- Big Match was my ~20% project there (:

# Intro

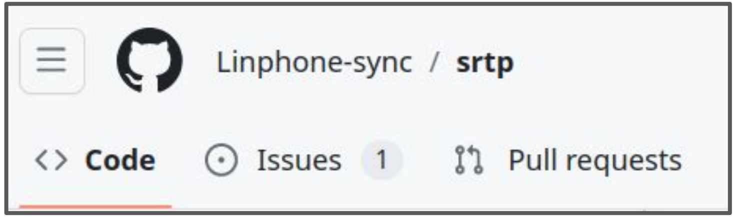
# Life of a Reverser



```
s_%s:_cloning_stream_(SSRC:_0x%08x_003338e8     XREF[1]:
    ds          "%s: cloning stream (SSRC: 0x%08x)\n"
```



"%s: cloning stream (SSRC: 0x%08x)\n"



Linphone-sync / srtp

# Other variations

- grep.app
- GitHub/Gitlab code search
- https://sourcegraph.com/search
- you name it
  - and tell me

# Why strings?

- Easy to see
- Easy to search for
- (mostly) **compiler-independent**
- (mostly) **platform-independent**
- Rarely change during a repo history

# Time for a story

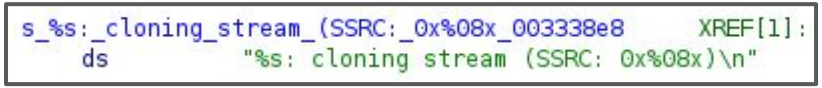
# It's 2018...

- Graduated from University
- **Need money**
- Don't want to help uncle with **grape harvesting**
- Somebody found trivial buffer overflows in Naver LINE's VoIP stack (**libAmp.so**)

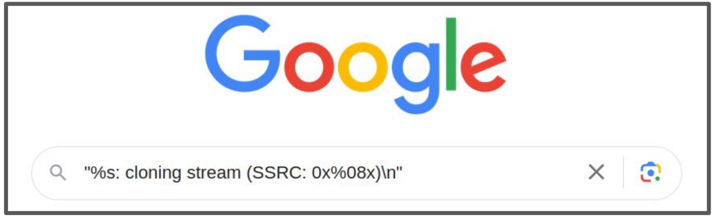
# It's 2018

- Graduate
- **Need mo**
- Don't war g
- Somebod ver LINE's Vo

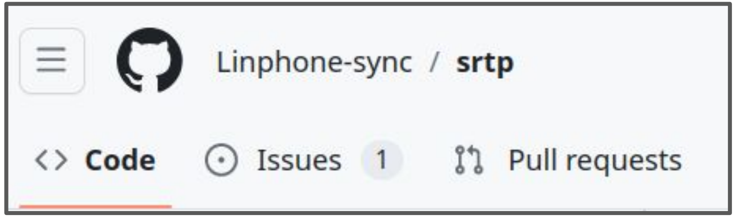
# Life of a Reverser (again)



```
s_%s:_cloning_stream_(SSRC:_0x%08x_003338e8      XREF[1]:
    ds          "%s: cloning stream (SSRC: 0x%08x)\n"
```

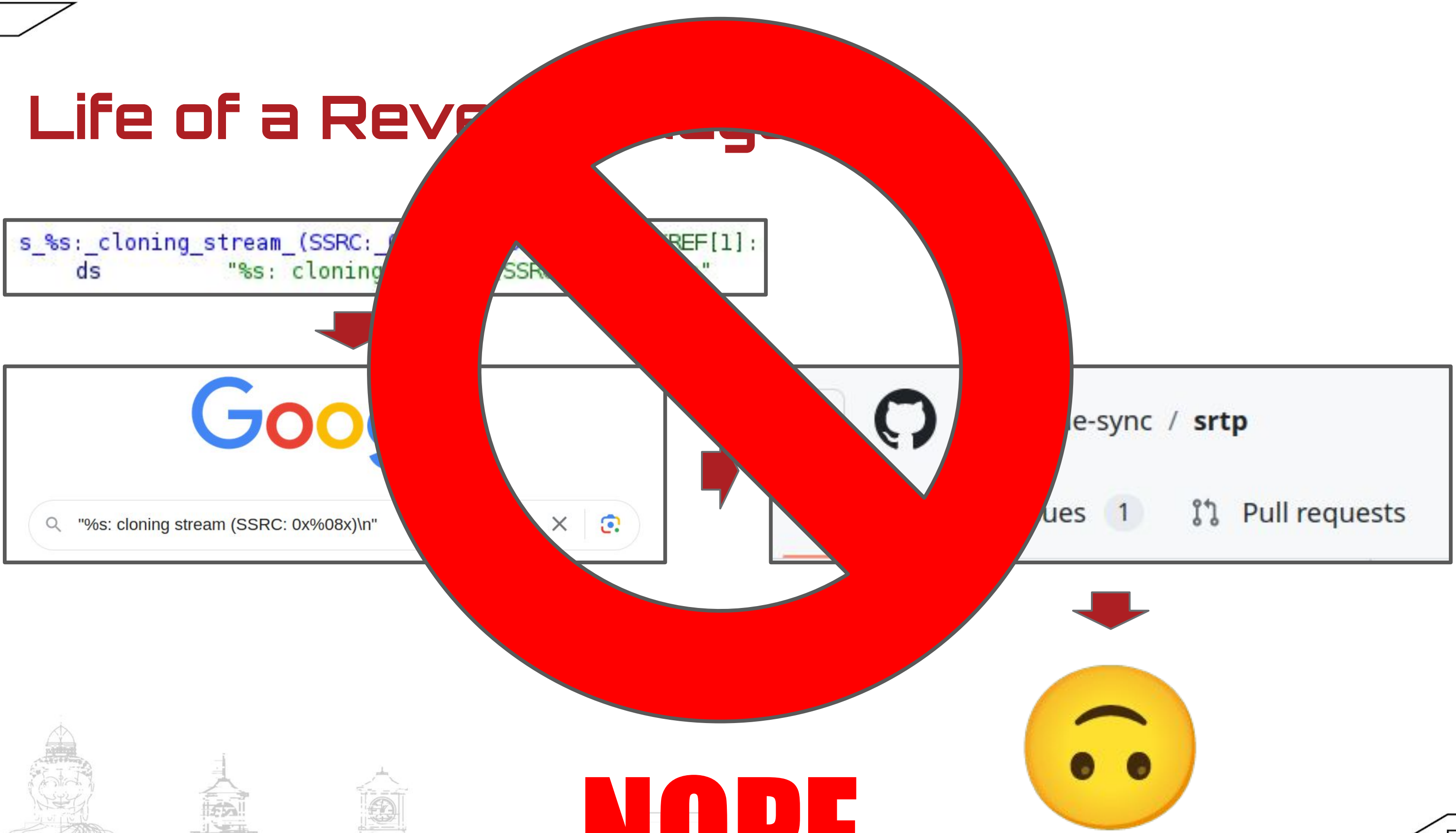


"%s: cloning stream (SSRC: 0x%08x)\n"



🙃

# Life of a Reve age

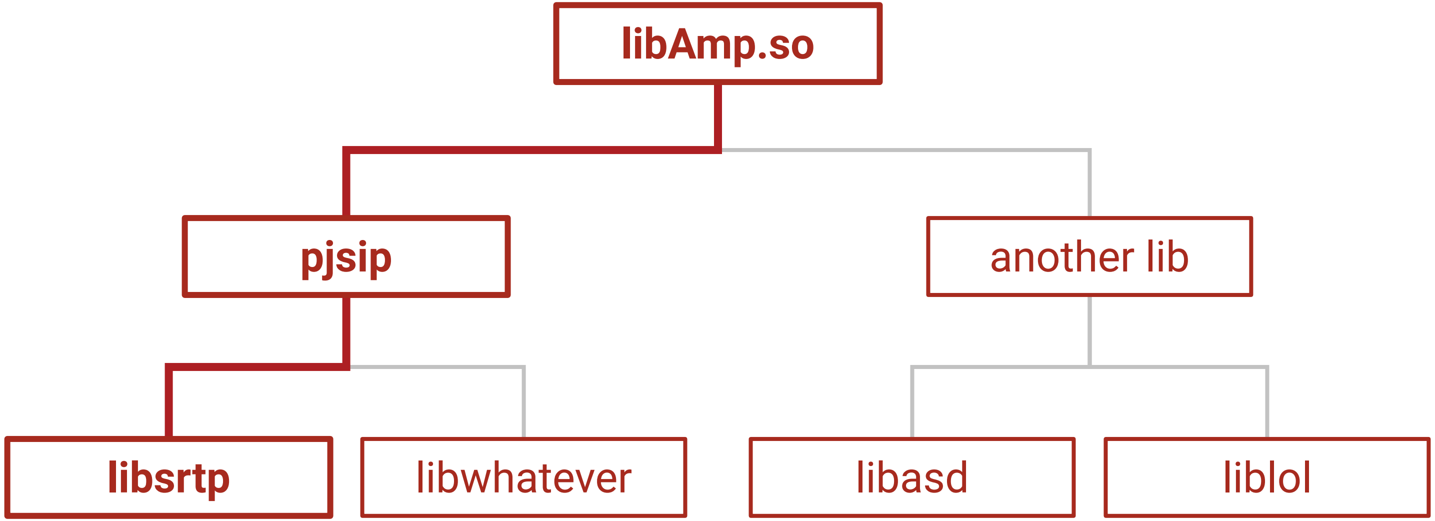


```
s_%s:_cloning_stream_(SSRC:_                    REF[1]:
    ds            "%s: cloning         SSR           "
```

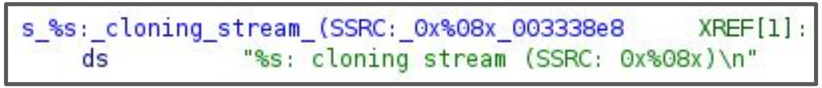
"%s: cloning stream (SSRC: 0x%08x)\n"

NOPE

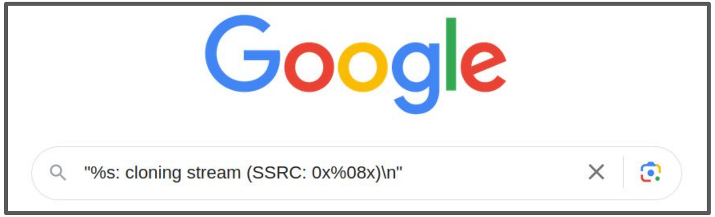
# Static libraries



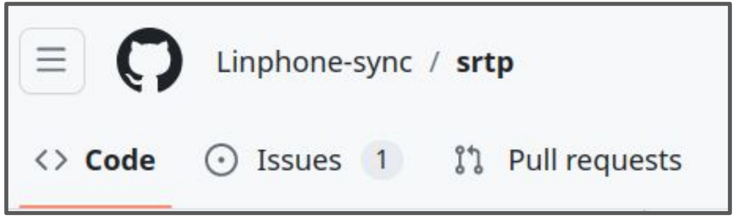
# Life of a Reverser (again)

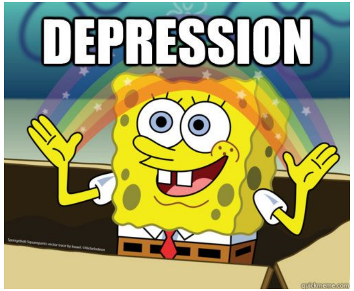


```
s_%s:_cloning_stream_(SSRC:_0x%08x_003338e8      XREF[1]:
     ds           "%s: cloning stream (SSRC: 0x%08x)\n"
```



"%s: cloning stream (SSRC: 0x%08x)\n"





# THE Problem

# Strings are not perfect

- Nested **statically-linked** libraries
- Parent libraries without strings
- **Not unique**
- Weird strings
  - **Hard to find**
- Google Search going A.I.
- Obfuscation (we will ignore this :D)

# THE Solution

# A huge Database of strings!

1. **Scan all C/C++ projects** on GitHub
2. Harvest strings
3. Throw ‘em into a Database
4. **Query** using target binary
5. …
6. PROFIT!!!

# Moar problems

# Not so easy

- How to download from GitHub at **scale**?
- **Parsing C/C++** is hard and slow
- Multiple **versions** on the same lib
- Projects with many **forks**
  - Linux kernel has 50k+ forks
- How do you **score** results?

# Not so easy (part 2)

- **Personal project**
- Limited resources (time, money, infra)
- KISS

# Our solution

# Outline

1. Get the source code of the **top-N C/C++ repositories** on GitHub (top ~ most starred)
2. **Deduplicate** the repositories
3. Extract the **strings**
4. **De-escape** the strings ('\n' => newline)
5. **Hash** the strings
6. Store them in some kind of database
7. Query the database using strings from target
8. **Cluster** the query results

# Dataset

# Getting the top-N repos

- Query GitHub API for projects
- Sort by **most starred**
- Clone them

# Getting the to

- Query GitHu jects
- Sort by **mos rred**
- Clone them

**RATE-LIMITING**

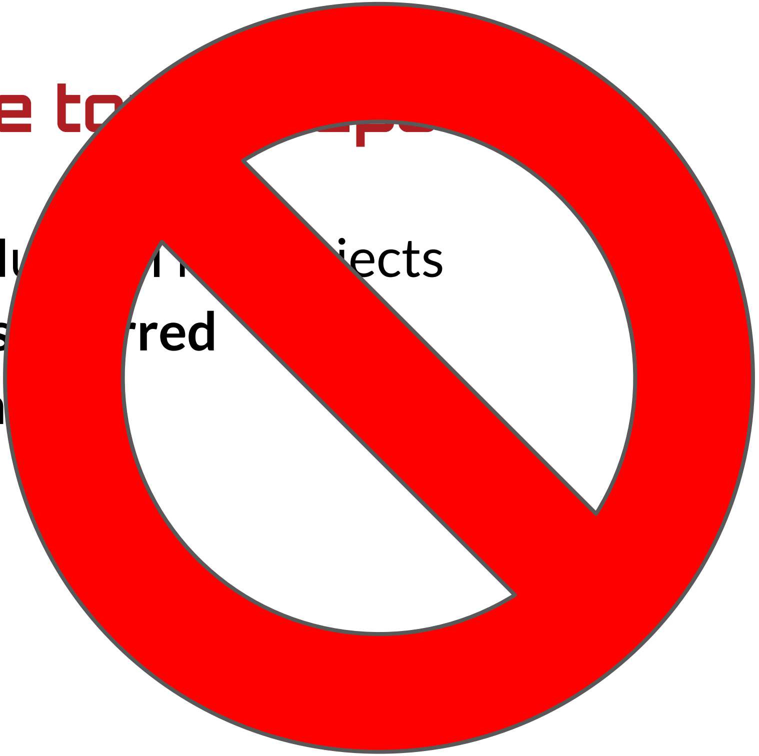
# Getting the to

- Query GitHu jects
- Sort by **mos** **rred**
- Clone them

**BANDWIDTH**

# Getting the to

- Query GitHu jects
- Sort by **mos rred**
- Clone them

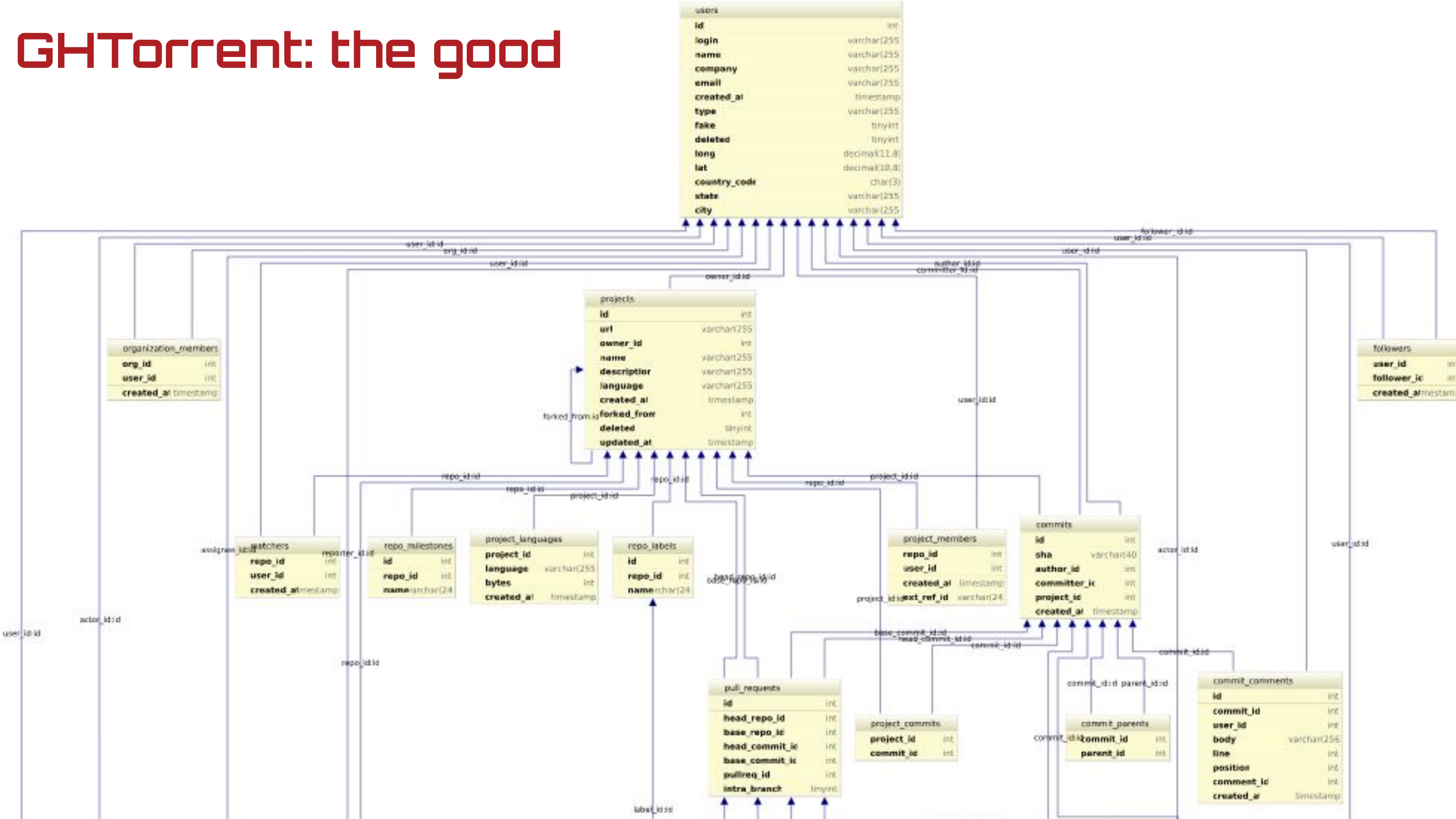


TIME

# GHTorrent

- Aka GitHub Torrent
  - Started in 2012
  - Prof. Georgios Gousios @ TU Delft
- **Polls GitHub** public events API
- Analyzes events
- Creates a **relational-view** of GH
- Available as **MySQL** or **MongoDB** dumps

# GHTorrent: the good

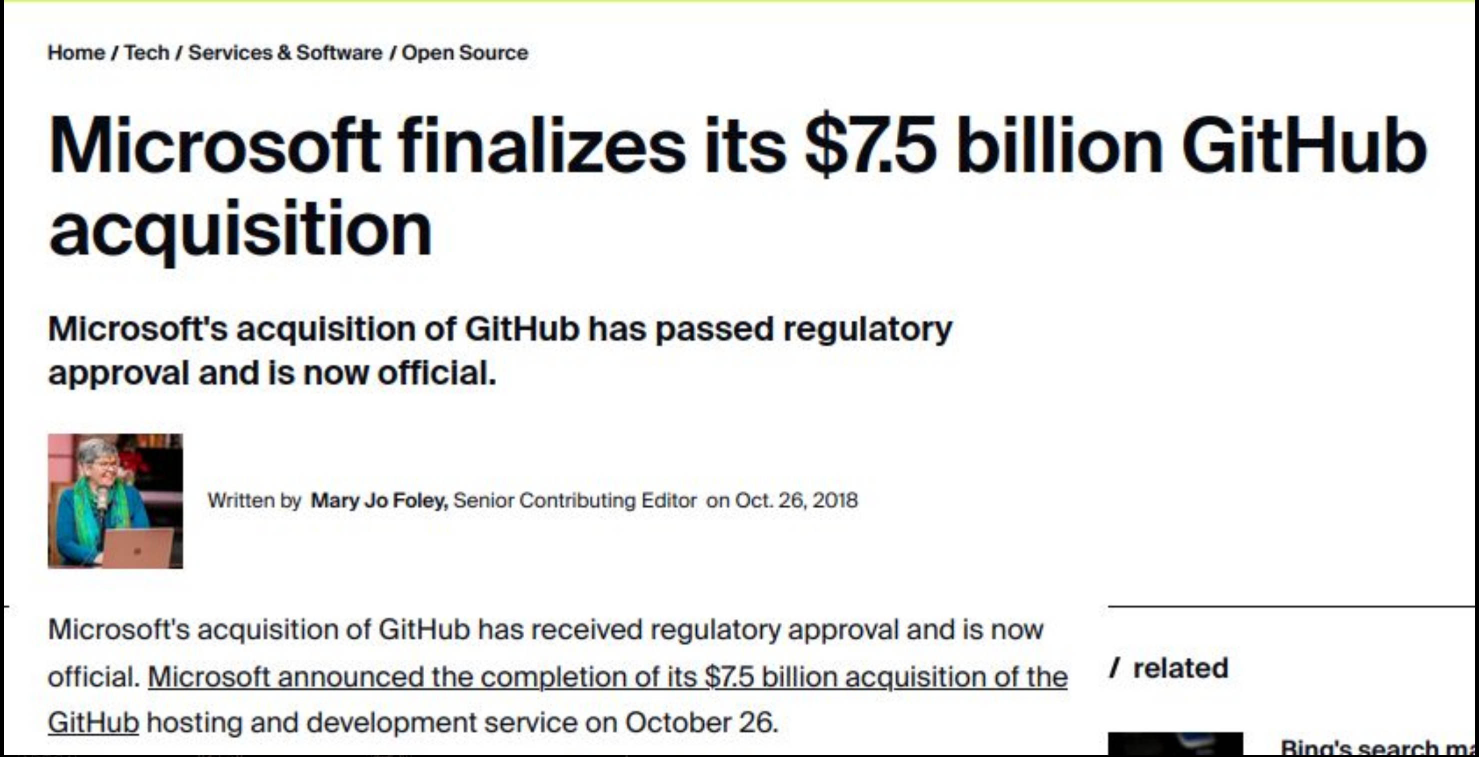


# GHTorrent: moar good

- You can **import** their dumps **locally**
- Query with SQL
- **Metadata:** projects, forks, stars, commits
- Most of the stuff we need 🙃

# GHTorrent: the bad

- **Best-effort**
  - Partial commit history
  - Missing/outdated data
- No source-code
- Looks like it’s **dead** 😞
  - This project requires $$$ and people
  - E.g.: Microsoft used to sponsor them



Home / Tech / Services & Software / Open Source

# Microsoft finalizes its $7.5 billion GitHub acquisition

## Microsoft's acquisition of GitHub has passed regulatory approval and is now official.

Written by **Mary Jo Foley,** Senior Contributing Editor on Oct. 26, 2018

Microsoft's acquisition of GitHub has received regulatory approval and is now official. Microsoft announced the completion of its $7.5 billion acquisition of the GitHub hosting and development service on October 26.

/ related

Bing's search ma

https://www.zdnet.com/article/microsoft-finalizes-its-7-5-billion-github-acquisition/

# GHTorrent: mongo

- **SQL** was used in the exploratory phase
  - We didn’t need all the tables
  - Too **slow** 👎
- Custom python tool
  - **bson** dumps
  - pymongo’s **`bson.decode_iter`**
  - Get info about projects, forks, and commits
  - **Fast** 👍

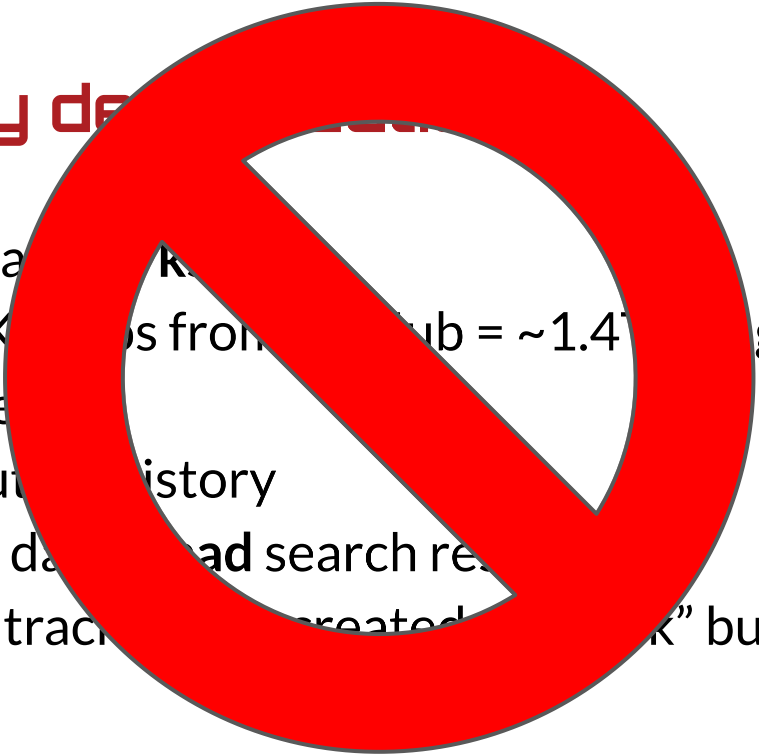
# Repo Deduplication

# Repository deduplication

- We don't want **forks**
- First ~100K repos from GitHub = ~1.4TB of gzip'd source code
    - Without git history
- **Duplicated** data = **bad** search results
- GHTorrent tracks forks created w/ “Fork” button

# Repository de

- We don't wa **k**
- First ~100K s from ub = ~1.4T gzip'd source code
  - ○ Without istory
- **Duplicated** da **ad** search re
- GHTorrent trac created K" button



**SADFACE.JPG**

# Repository deduplication: ++problems

- How do you define a project?
- How do you define repo A is a duplicate of repo B?
- How about popular monorepos?
  - https://github.com/freebsd/freebsd-src

We decided to use **root commits*** + **custom algorithm**

*root commit = first commit in repo history

# Workaround: git history

With infinite resources:

1. Clone a repo
2. Put every commit in a graph DB
3. Connect commits using **parent/child** relationship
4. Repeat 1-3 until you are done, then…
5. Look for **root commits**
6. For each root commit, keep the **most-starred repo**

# Workaround workaround: GHTorrent

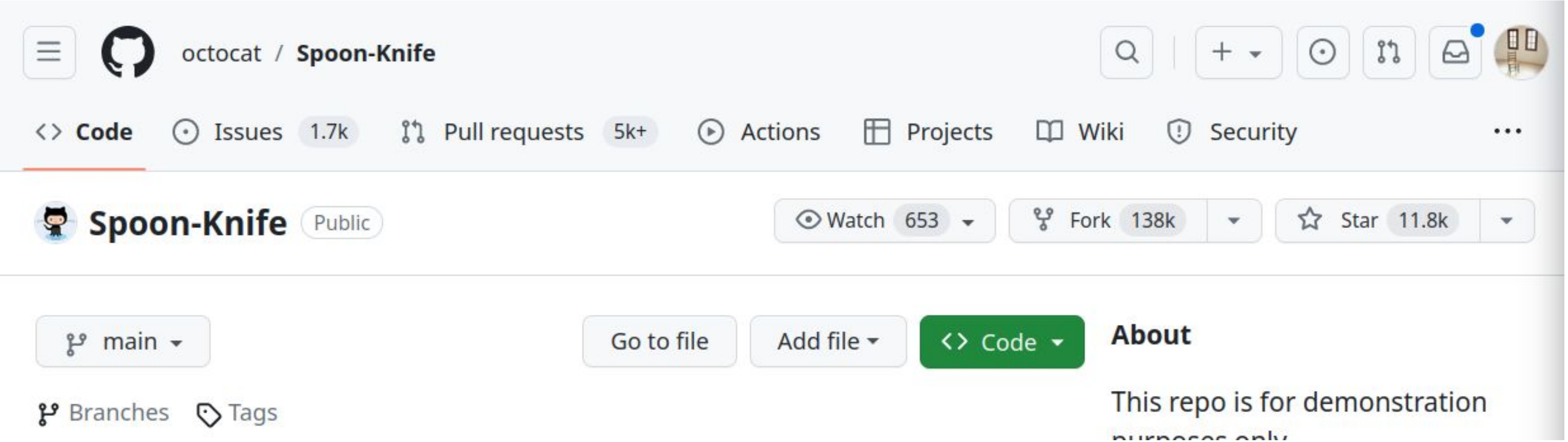
- Deduplicate **before** cloning
  - Best effort
- Strike a balance
  - Deduplicate enough => only **keep good stuff**
  - Don't over-do it => **remove** only **bad stuff**

We thought we had a perfect solution but…

# Repository deduplication: ++problems

People do **weird s***t** with their git history.

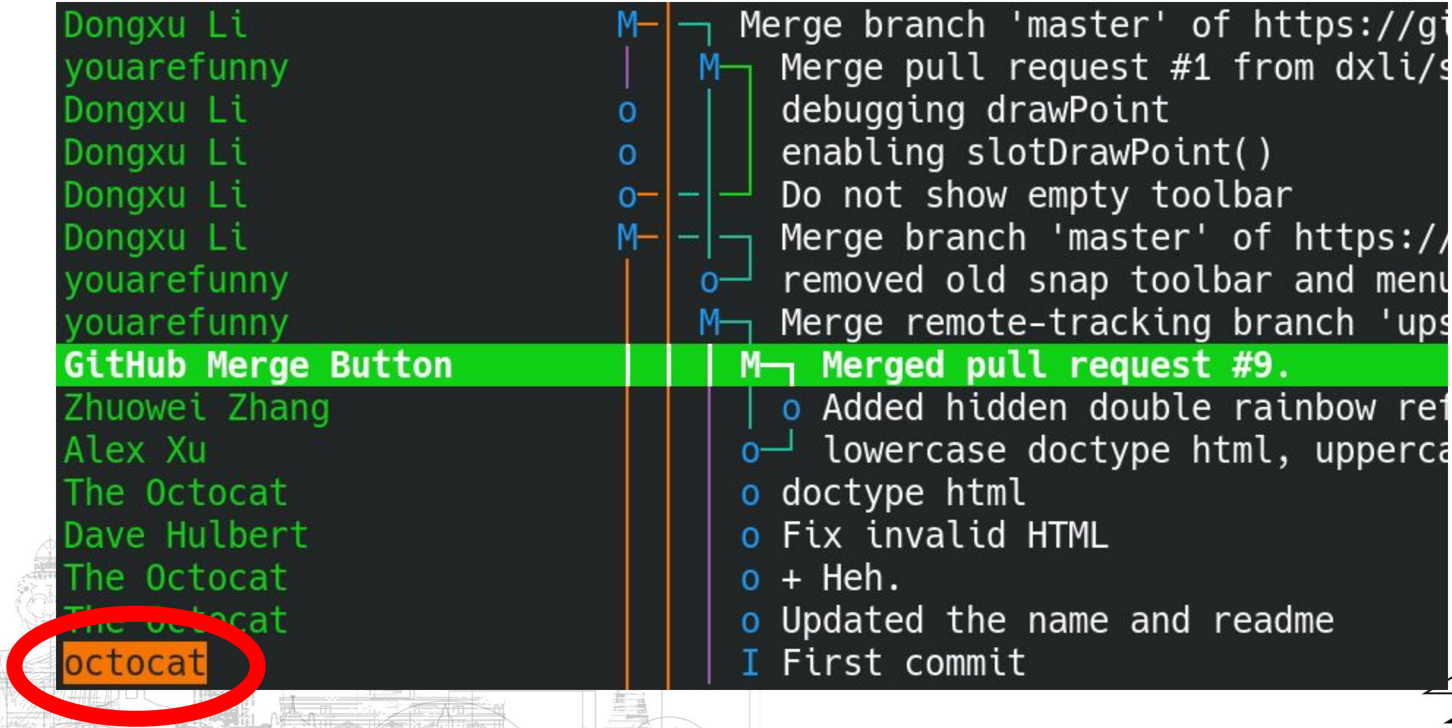
# Story time 2



About

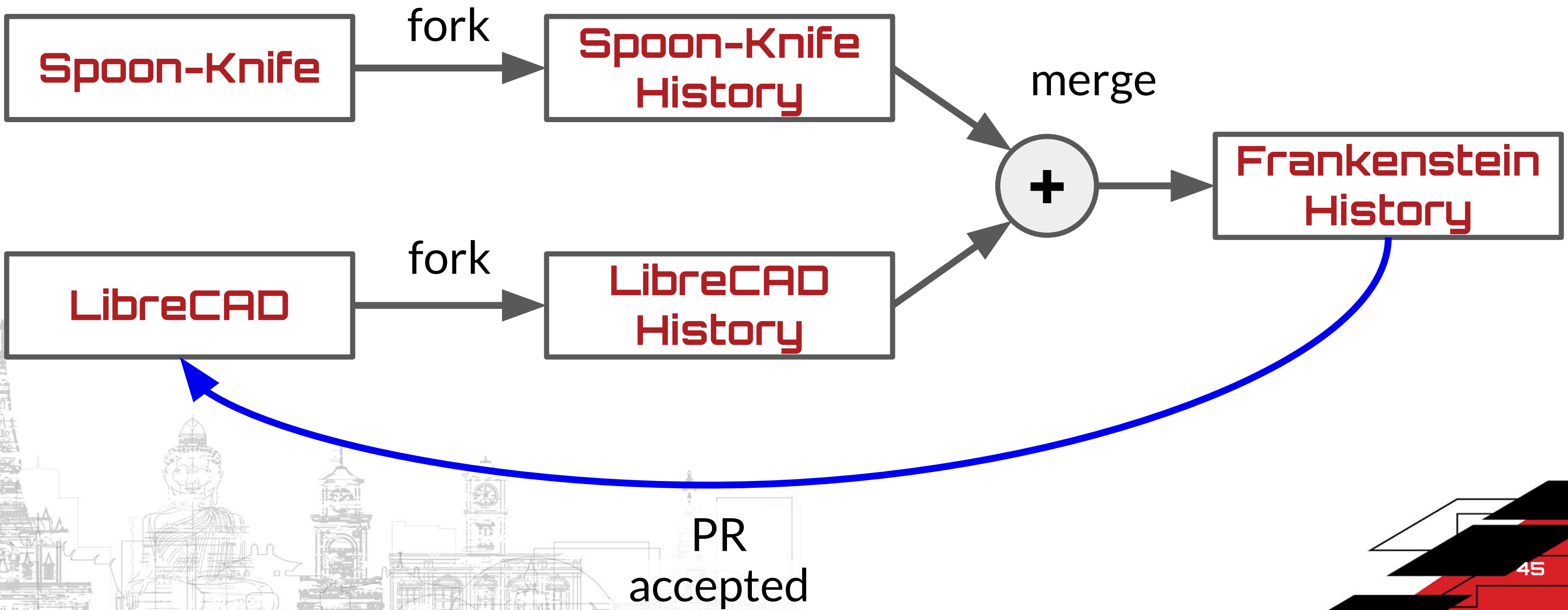
This repo is for demonstration purposes only

# Story time 2: LibreCAD



```
Dongxu Li              M─ ─┐ Merge branch 'master' of https://gi
youarefunny            │    M─┐ Merge pull request #1 from dxli/s
Dongxu Li              o    │ │ debugging drawPoint
Dongxu Li              o    │ │ enabling slotDrawPoint()
Dongxu Li              o─ ─│─┘ Do not show empty toolbar
Dongxu Li              M─ ─│─┐ Merge branch 'master' of https://
youarefunny            │    o─┘ removed old snap toolbar and menu
youarefunny            │    M─┐ Merge remote-tracking branch 'ups
GitHub Merge Button    │ │  │ M─┐ Merged pull request #9.
Zhuowei Zhang          │ │  │ │ o Added hidden double rainbow ref
Alex Xu                │ │  │ o─┘ lowercase doctype html, upperca
The Octocat            │ │  │ o doctype html
Dave Hulbert           │ │  │ o Fix invalid HTML
The Octocat            │ │  │ o + Heh.
The Octocat            │ │  │ o Updated the name and readme
octocat                │ │  │ I First commit
```

# Story time 2: user “youarefunny”



Spoon-Knife

fork

Spoon-Knife History

merge

+

Frankenstein History

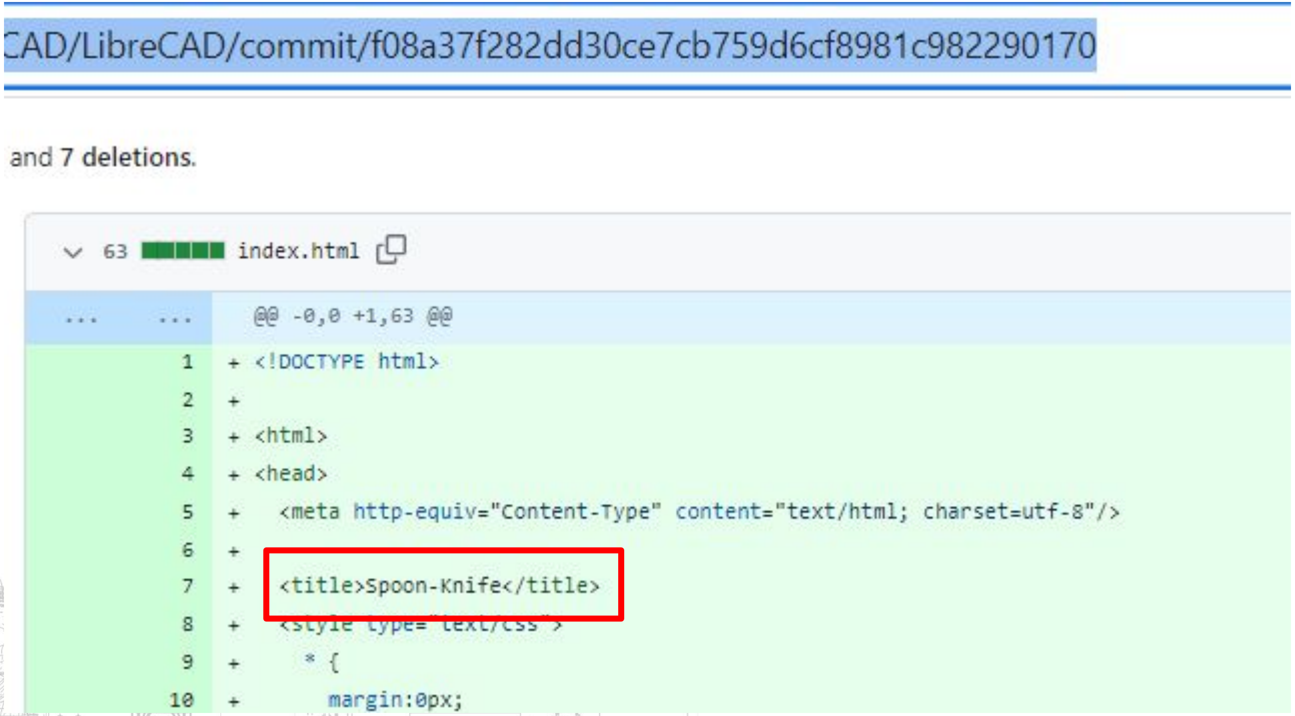
LibreCAD

fork

LibreCAD History

PR accepted

# If you don’t believe me



CAD/LibreCAD/commit/f08a37f282dd30ce7cb759d6cf8981c982290170

and 7 deletions.

```
@@ -0,0 +1,63 @@
+ <!DOCTYPE html>
+
+ <html>
+ <head>
+   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
+
+   <title>Spoon-Knife</title>
+   <style type="text/css">
+     * {
+       margin:0px;
```

https://github.com/LibreCAD/LibreCAD/commit/f08a37f282dd30ce7cb759d6cf8981c982290170

# What's the problem?

- LibreCAD now has **2 root commits**
- Spoon-Knife has **more stars**

=> Our algo throws away LibreCAD

## What's the pr

- LibreCAD w 2 **root con ts**
- Spoon-K has **m tars**

=> Our algo t ws away L C

I HATE LIFE

# Best-effort deduplication

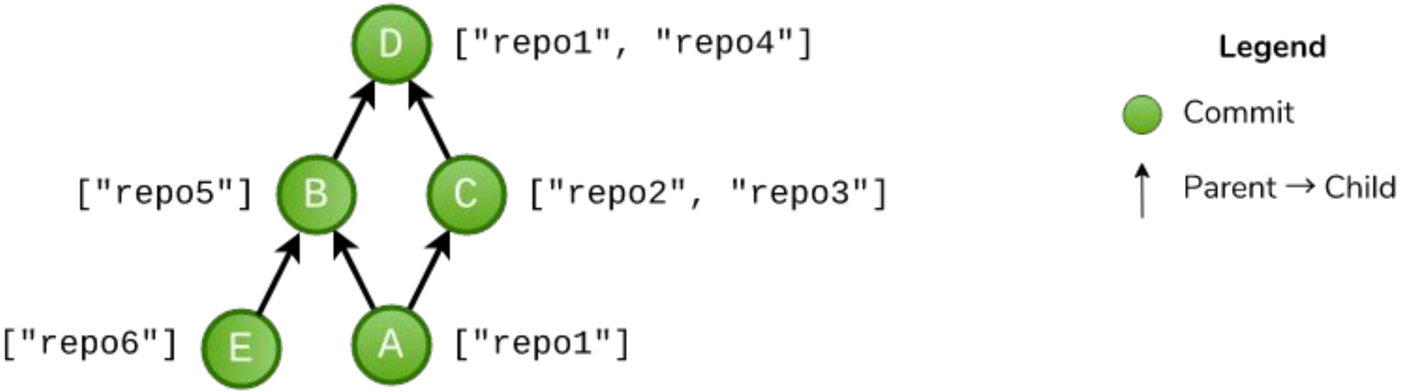
- GHTorrent
  - (**parent** commit, **child** commit) partial relations
  - (**commit**, **repo**) partial relations

# Best-effort deduplication: the algo

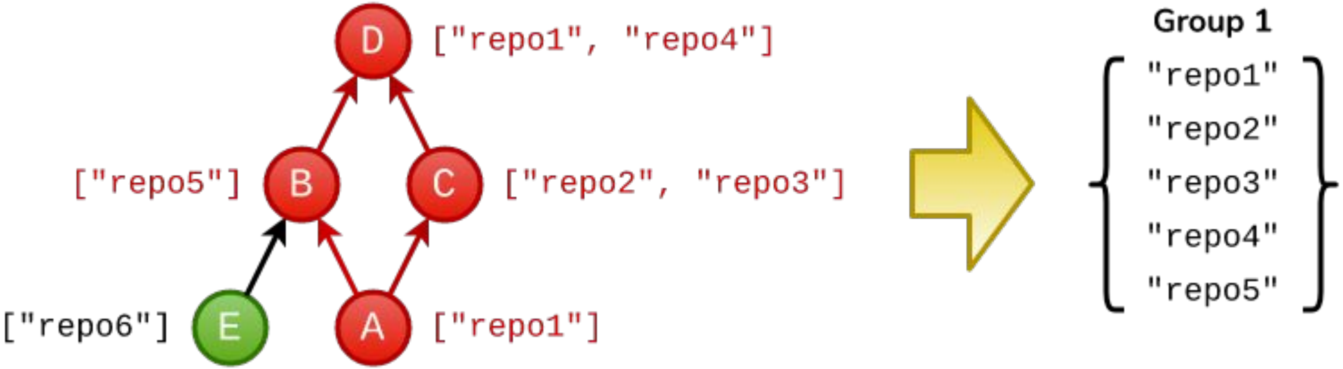
1. Find commit **without parents** (*parentless commit*)
2. Create a history subgraph following **parent => child** edges
3. Group all repos associated with the commits from 2 (*repository group*)
4. For every group, the **most starred repo** will be considered a parent, the others will be children
   - We have **parent repo => child repo** edges now
5. Do 1-4 for every repo, create huge graph of **parent/child repos**
6. Only crawl **repos without a parent**

# I know that was hard

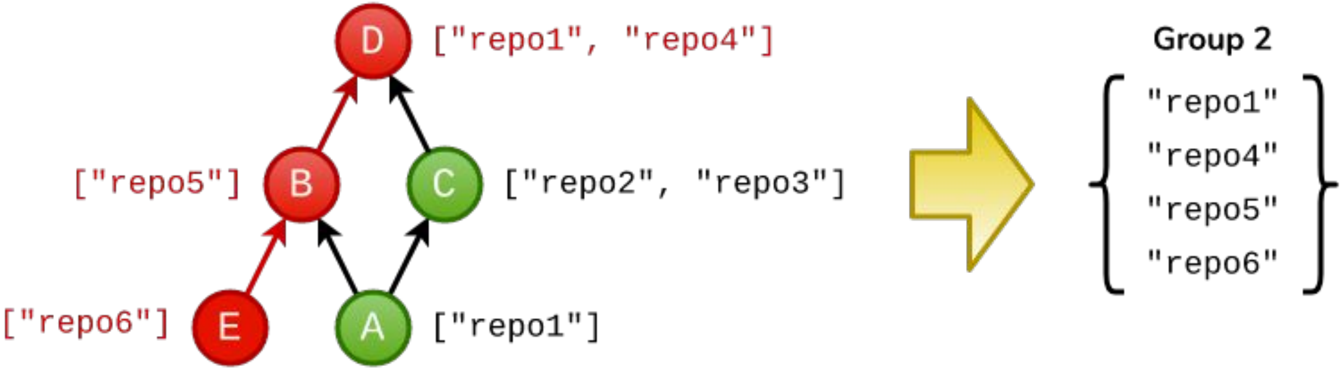
# Deduplication example



# Repo group 1


D
["repo1", "repo4"]
["repo5"]
B
C
["repo2", "repo3"]
["repo6"]
E
A
["repo1"]
Group 1
"repo1"
"repo2"
"repo3"
"repo4"
"repo5"

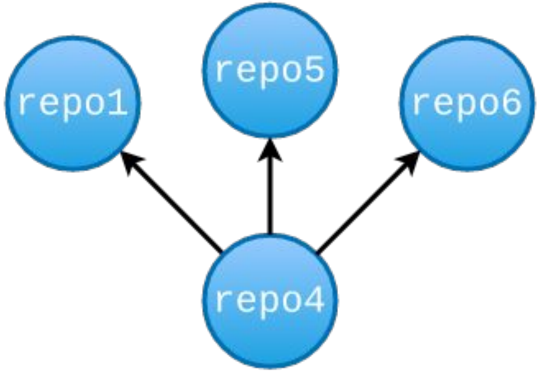
# Repo group 2


D
["repo1", "repo4"]
["repo5"]
B
C
["repo2", "repo3"]
["repo6"]
E
A
["repo1"]
Group 2
"repo1"
"repo4"
"repo5"
"repo6"

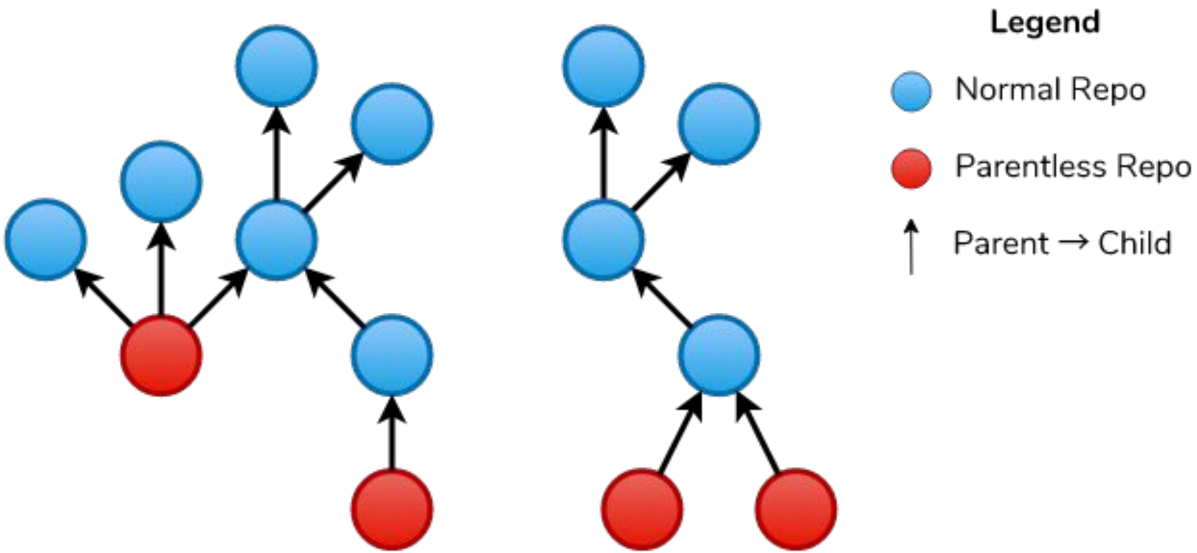
# Partial repo graph

**Group with stars**

```
{
  "repo1": 11 ★
  "repo4": 50 ★
  "repo5": 20 ★
  "repo6": 10 ★
}
```



# Repo graph



# Deduplication: full disclosure

- I know our algo is **not perfect**
- We found it has a good **balance**

# Processing repos

# Extracting strings

- **Parsing** C/C++ files is **non-trivial**
  - macros, includes, other black magic
- We wanted a fast PoC
  - ripgrep

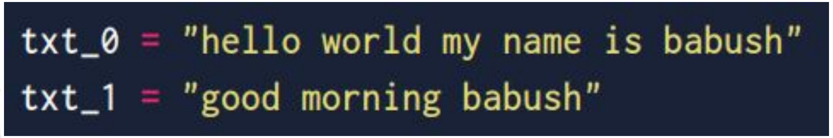
# Processing strings

- **De-escape**, aka ‘\n’ => byte 0x0A
  - noescape
  - https://github.com/thebabush/noescape
- Hash
  - sha256

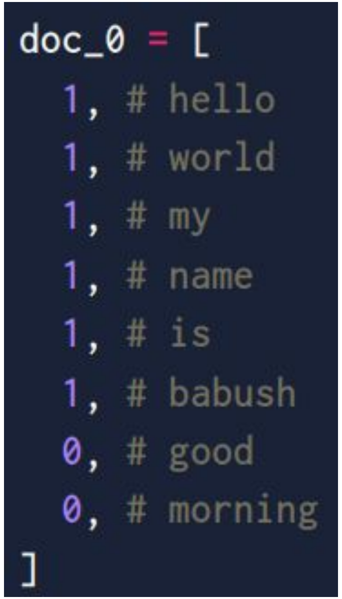
# Polishing the data

# Search engine 101

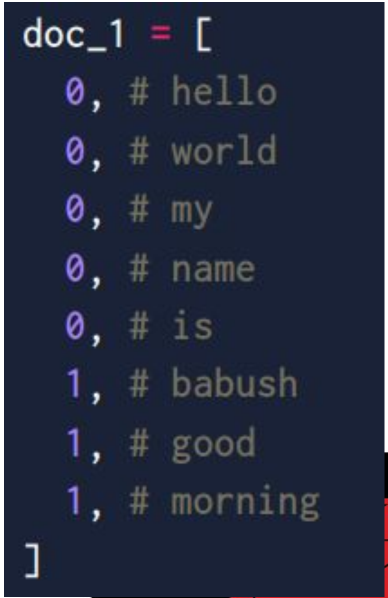
- **Vector-space model**
- Score = **similarity** between vectors



```
txt_0 = "hello world my name is babush"
txt_1 = "good morning babush"
```



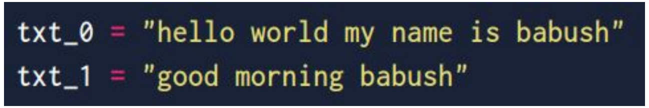
```
doc_0 = [
  1, # hello
  1, # world
  1, # my
  1, # name
  1, # is
  1, # babush
  0, # good
  0, # morning
]
```



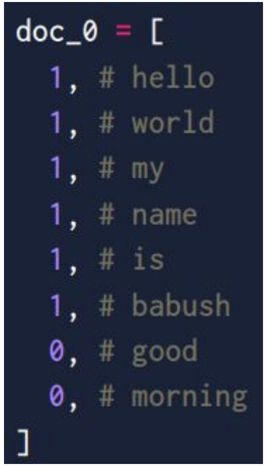
```
doc_1 = [
  0, # hello
  0, # world
  0, # my
  0, # name
  0, # is
  1, # babush
  1, # good
  1, # morning
]
```

# Why?

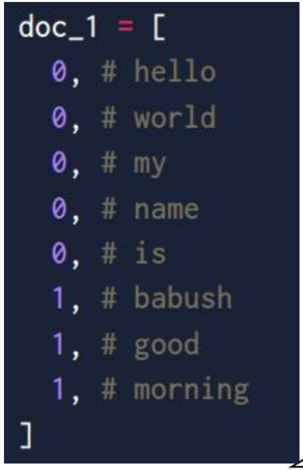
- Swap documents with repositories
- Swap words with string hashes



```
txt_0 = "hello world my name is babush"
txt_1 = "good morning babush"
```



```
doc_0 = [
  1, # hello
  1, # world
  1, # my
  1, # name
  1, # is
  1, # babush
  0, # good
  0, # morning
]
```



```
doc_1 = [
  0, # hello
  0, # world
  0, # my
  0, # name
  0, # is
  1, # babush
  1, # good
  1, # morning
]
```

# Building a robust data pipeline

- Needed a **fast** and **solid** pipeline
- We went with the usual data-science frameworks

aws

Apache Airflow

TensorFlow

APACHE hadoop

APACHE Spark™

# Building a rob ne

- Needed a f in d pipeline
- We went w the us ta-scien framewor

Apache Airflow

TensorFlow

oop

aws

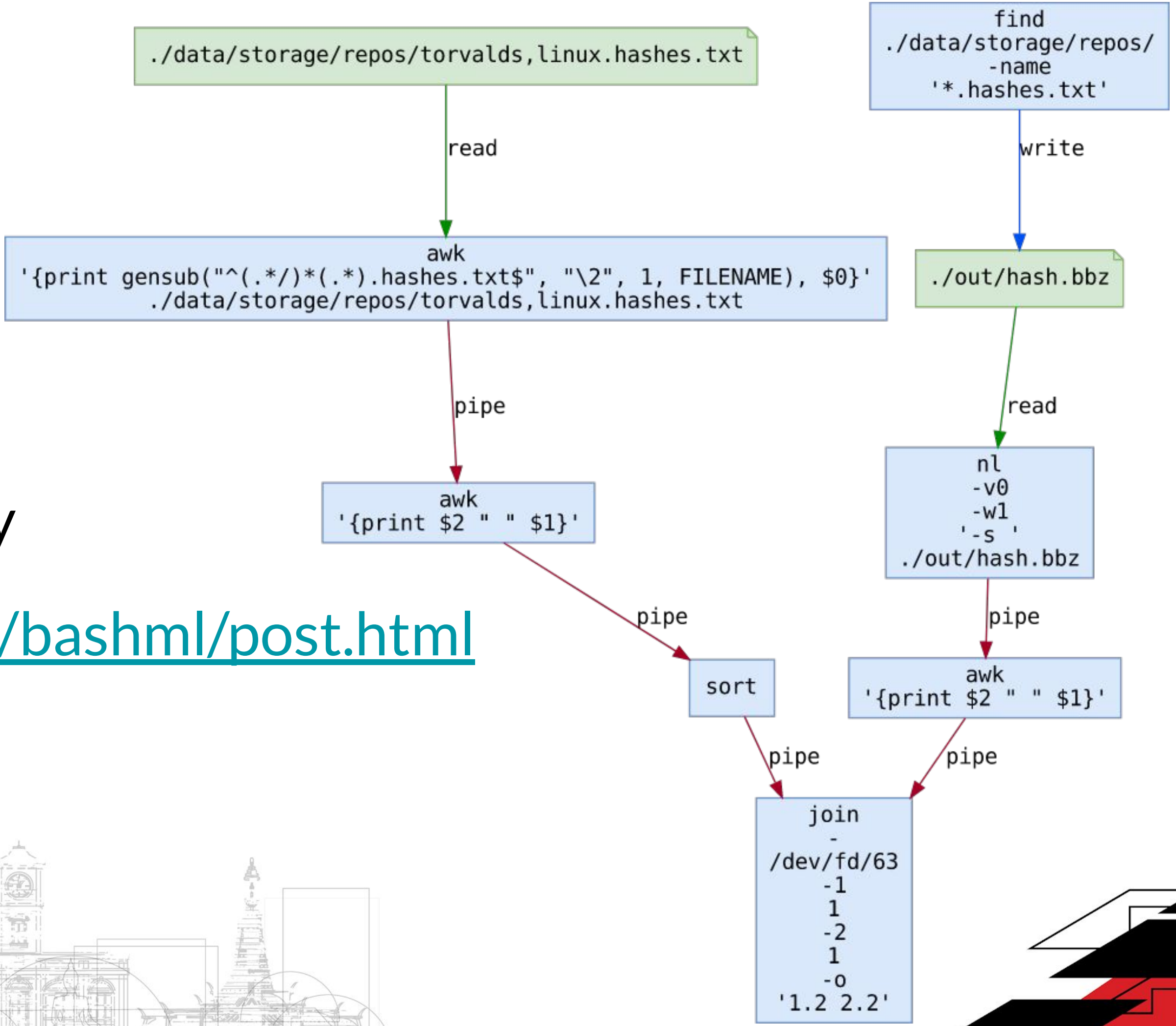
NOPE

APACHE Spark™

# Bash

It’s bash all the way

https://rev.ng/blog/bashml/post.html



```
./data/storage/repos/torvalds,linux.hashes.txt

find
./data/storage/repos/
-name
'*.hashes.txt'

awk
'{print gensub("^(.*/)*(.*).hashes.txt$", "\2", 1, FILENAME), $0}'
./data/storage/repos/torvalds,linux.hashes.txt

./out/hash.bbz

awk
'{print $2 " " $1}'

nl
-v0
-w1
'-s '
./out/hash.bbz

sort

awk
'{print $2 " " $1}'

join
-
/dev/fd/63
-1
1
-2
1
-o
'1.2 2.2'
```

# Algorithms, parameter estimation

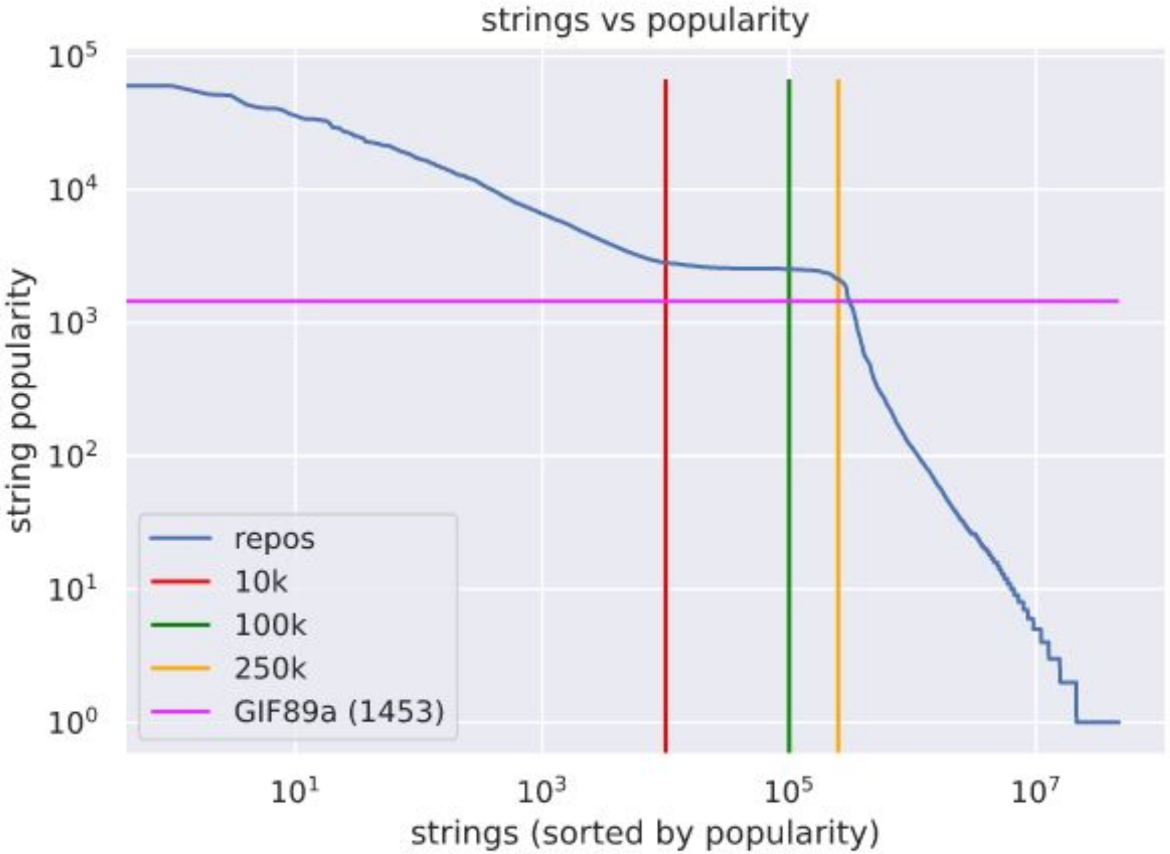
Vector-space model requires some choices

- Built a **synthetic dataset** using Gentoo
    - Statically link many packages
    - Use it as ground truth
- Results
    - Weighting: **tf-idf**
    - Scoring: **cosine similarity**

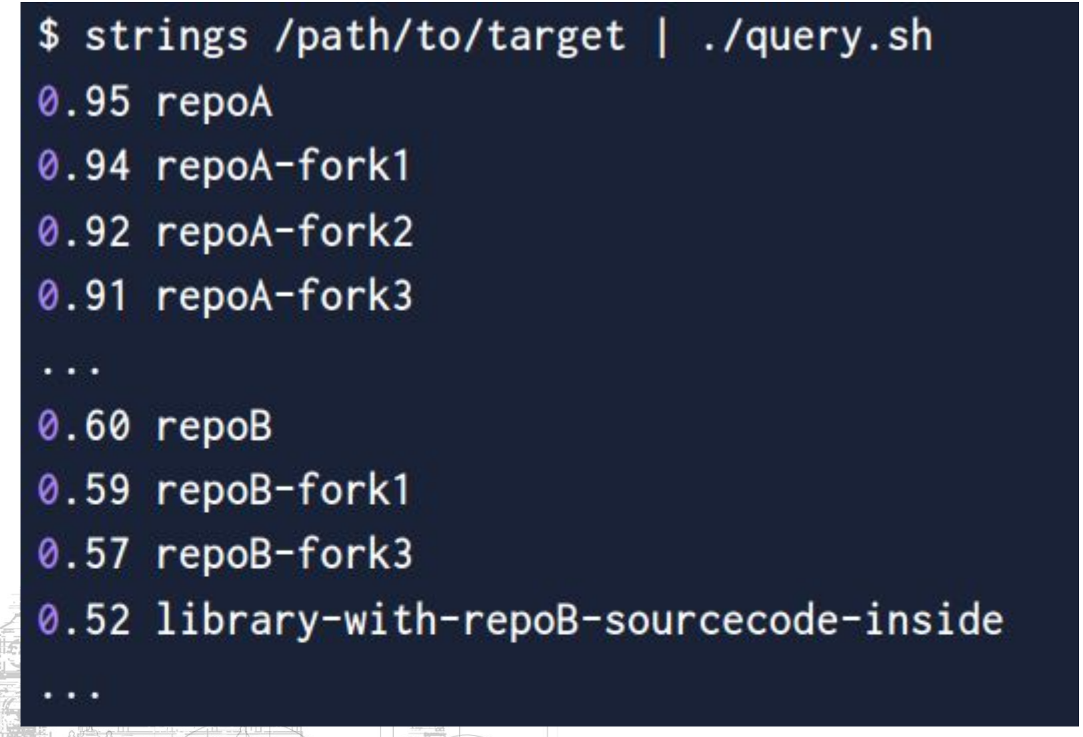
# Removing useless strings

- Did some tests on synthetic dataset
    - **Common** strings are **bad** (lower accuracy)
        - e.g.: “error”, etc…
    - Removed **top 10K** most popular strings

# Removing useless strings



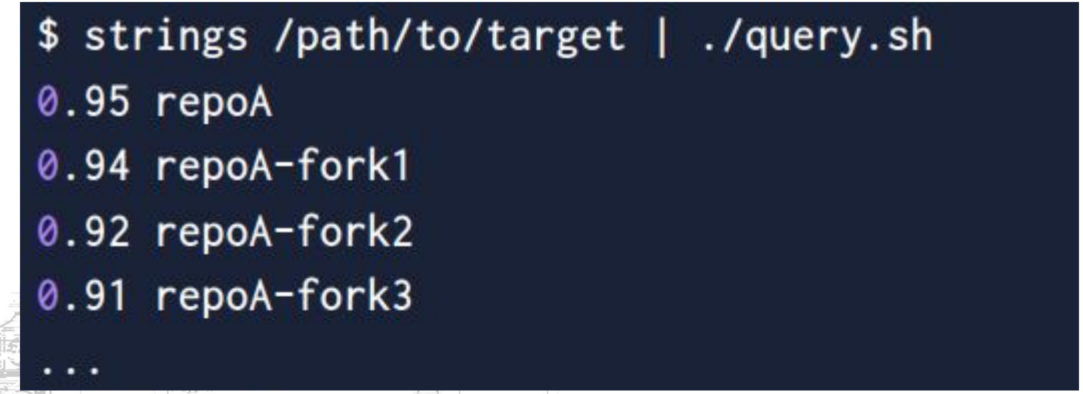
# Still bad results :(



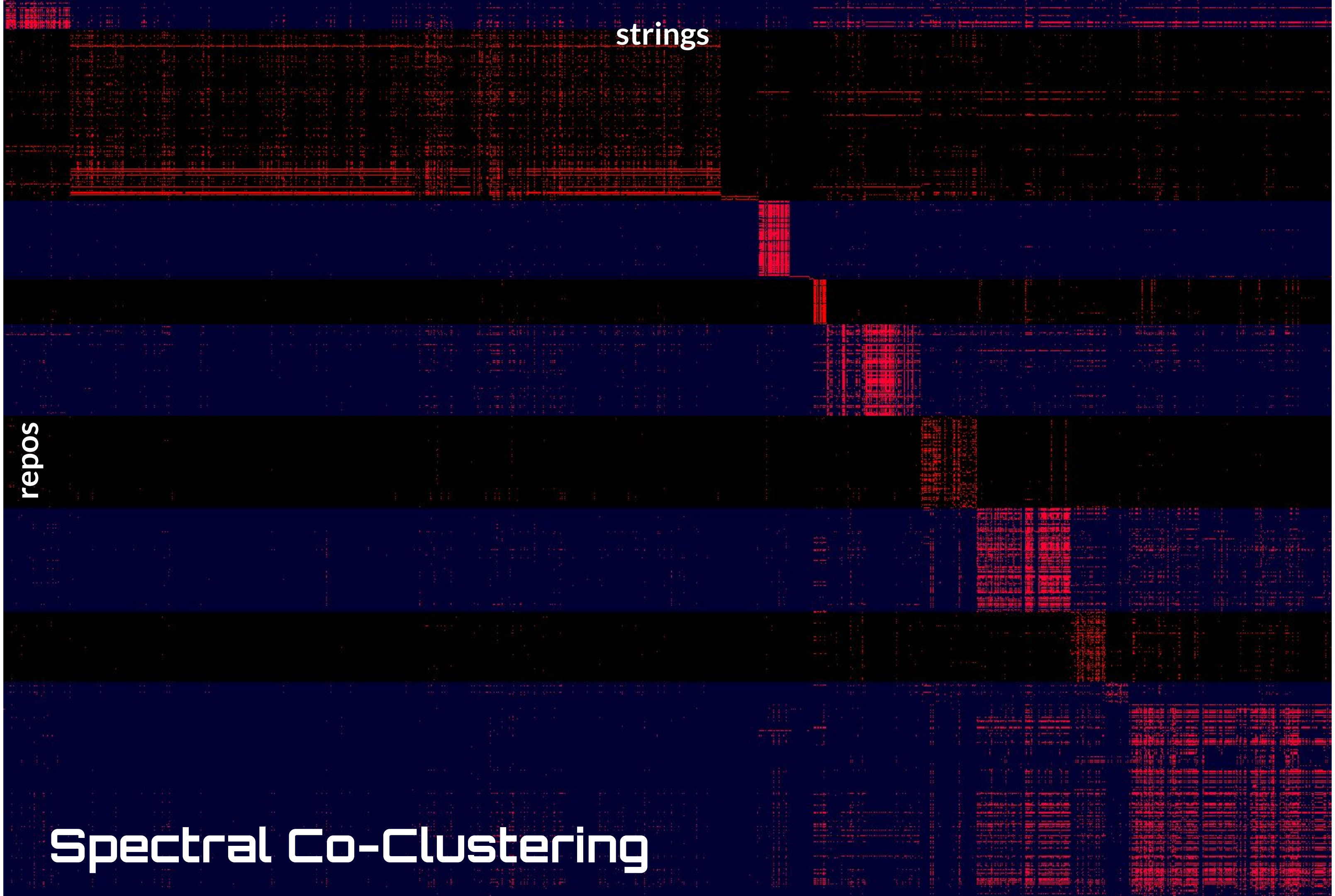
```
$ strings /path/to/target | ./query.sh
0.95 repoA
0.94 repoA-fork1
0.92 repoA-fork2
0.91 repoA-fork3
...
0.60 repoB
0.59 repoB-fork1
0.57 repoB-fork3
0.52 library-with-repoB-sourcecode-inside
...
```

# Still bad results :(

- Let’s say a target uses **zlib** and **libssl**
- One of the two will be **buried in the results**
    - Both libs have many forks/duplicates



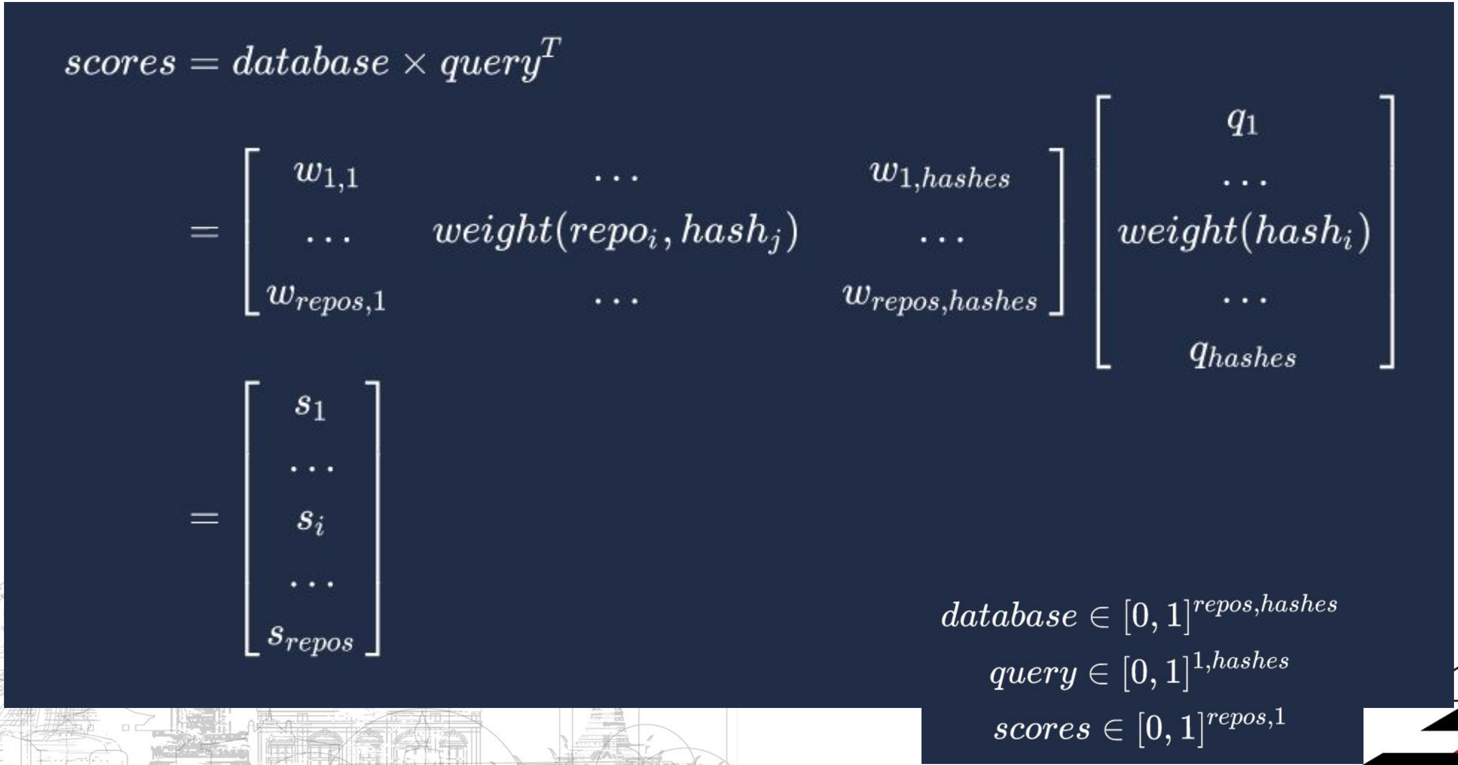
```
$ strings /path/to/target | ./query.sh
0.95 repoA
0.94 repoA-fork1
0.92 repoA-fork2
0.91 repoA-fork3
...
```



# Spectral Co-Clustering

# Putting everything into production

# Python + Sparse Matrices



$$scores = database \times query^T$$

$$= \begin{bmatrix} w_{1,1} & \dots & w_{1,hashes} \\ \dots & weight(repo_i, hash_j) & \dots \\ w_{repos,1} & \dots & w_{repos,hashes} \end{bmatrix} \begin{bmatrix} q_1 \\ \dots \\ weight(hash_i) \\ \dots \\ q_{hashes} \end{bmatrix}$$

$$= \begin{bmatrix} s_1 \\ \dots \\ s_i \\ \dots \\ s_{repos} \end{bmatrix}$$

$$database \in [0,1]^{repos,hashes}$$

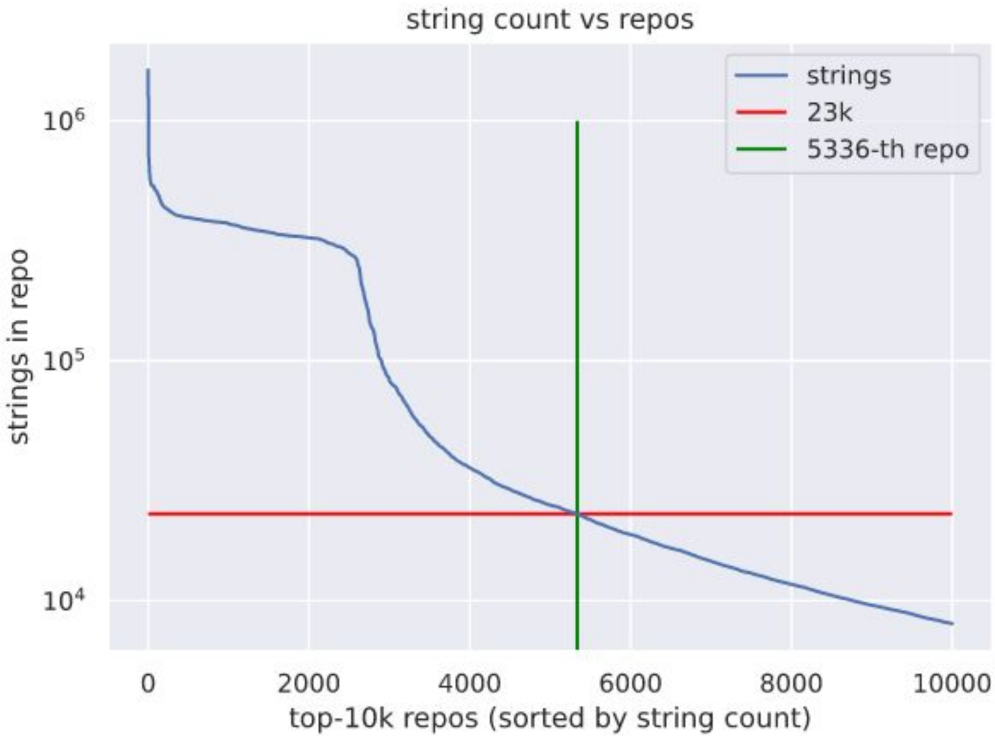
$$query \in [0,1]^{1,hashes}$$

$$scores \in [0,1]^{repos,1}$$

# Moar deduplication

# Resources = $$$

- Avg **RAM** per repo **~40kB**
- Avg **string count ~23k**



# Second dedup algo

- Take a repo
- Look for K repos of **similar size**
- If (jaccard_similarity(A, B) > threshold) => delete B
- Complexity **O(N * K)**

Takes care of a lot of linux/Android/etc source dumps.

# DEMO
# https://bigmatch.rev.ng

# Almost done,
# I promise

# Pros

- **Perfect string-matching** works surprisingly well
- **Privacy**
  - if a hash doesn't match, we don't know what string it represents
- **0%** machine learning

# Cons

- Only works for targets with **good strings**
- No **partial matching**
- Query speed good
  - But this is a PoC-sized DB
- `strings` is not very good
  - Wrong prefixes (e.g.: "XRWFHello World")
  - Better **use a decompiler** to extract strings

# Future

- Integrate Big Match with rev.ng **decompiler**
- **Partial** string matching
- Support **magic numbers**/arrays
- Use strings to guess library version-range
- Add strings from decompiled firmwares/etc
- Actually parse C/C++ files
  - E.g.: per-function strings
  - I actually have a demo of this (:

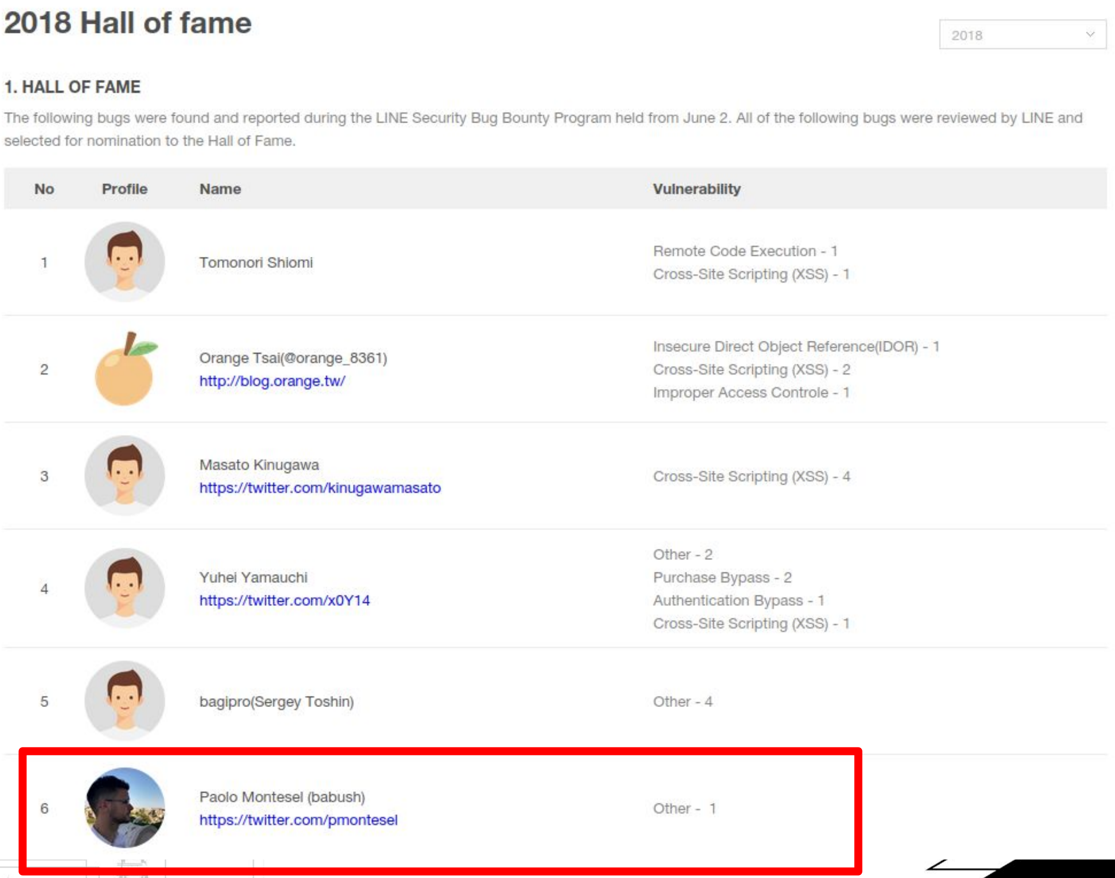
# Some other applications

- Figure out which libraries are used in a **monorepo**
  - Find vulnerable deps that GH doesn’t catch :D
- **Malware** classification
- Other languages

# Happy ending

No grape harvesting

w/ uncle



## 2018 Hall of fame

2018

### 1. HALL OF FAME

The following bugs were found and reported during the LINE Security Bug Bounty Program held from June 2. All of the following bugs were reviewed by LINE and selected for nomination to the Hall of Fame.

| No | Profile | Name | Vulnerability |
|---|---|---|---|
| 1 | | Tomonori Shiomi | Remote Code Execution - 1<br>Cross-Site Scripting (XSS) - 1 |
| 2 | | Orange Tsai(@orange_8361)<br>http://blog.orange.tw/ | Insecure Direct Object Reference(IDOR) - 1<br>Cross-Site Scripting (XSS) - 2<br>Improper Access Controle - 1 |
| 3 | | Masato Kinugawa<br>https://twitter.com/kinugawamasato | Cross-Site Scripting (XSS) - 4 |
| 4 | | Yuhei Yamauchi<br>https://twitter.com/x0Y14 | Other - 2<br>Purchase Bypass - 2<br>Authentication Bypass - 1<br>Cross-Site Scripting (XSS) - 1 |
| 5 | | bagipro(Sergey Toshin) | Other - 4 |
| 6 | | Paolo Montesel (babush)<br>https://twitter.com/pmontesel | Other - 1 |

https://bugbounty.linecorp.com/en/halloffame/2018/

# Moar slides

# JK

# THANK YOU!

# Questions?

- https://rev.ng/blog/big-match/post.html
- https://bigmatch.rev.ng
- http://www.babush.me/

babushkam

pmontesel