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 \(\cap\) data-science \(\cap\) 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)
```

Google search:
```
"%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.

- Graduated
- Need money
- Don’t want to help uncle with grape harvesting
- Somebody found trivial buffer overflows in Naver LINE’s VoIP stack

STOP!

BOUNTY TIME!
Life of a Reverser (again)
Life of a Reverser (again)

NOPE
Static libraries

libAmp.so

- pjsip
  - libsrt
  - libwhatever
- another lib
  - libasd
  - liblol
Life of a Reverser (again)

```
s_%s: cloning_stream (SSRC: 0x%08x)_003338e8
  XREF[1]:
sds
  "%s: cloning stream (SSRC: 0x%08x)\n"
```

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

Linphone-sync / srtp

Pull requests

DEPRESSION
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 top-N repos:

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

**RATE-LIMITING**
Getting the top-N repos

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

**Bandwidth**
Getting the top-N repos

- Query GitHub API for projects
- Sort by most starred
- 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
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.
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 deduplication

- We don't want forks
- First ~100K repos from GitHub = ~1.4TB of gzip'd source code
  - Without history
- Duplicated data leads to bad search results
- GHTorrent tracks forks created via "Fork" button
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
Story time 2: LibreCAD

GitHub Merge Button
- Dongxu Li
- youarefunny
- Dongxu Li
- Dongxu Li
- youarefunny
- youarefunny
- Zhuowei Zhang
- Alex Xu
- The Octocat
- Dave Hulbert
- The Octocat
- The Octocat
- octocat

- Merge branch 'master' of https://github.com/dxli/shipyard
- Merge pull request #1 from dxli/shipyard
- debugging drawPoint
- enabling slotDrawPoint()
- Do not show empty toolbar
- Merge branch 'master' of https://github.com/dxli/shipyard
- removed old snap toolbar and menu
- Merge remote-tracking branch 'ups'
- Merged pull request #9.
- Added hidden double rainbow regex
- lowercase doctype html, uppercase html
- doctype html
- Fix invalid HTML
- + Heh.
- Updated the name and readme
- First commit
Story time 2: user “youarefunny”

Spoon-Knife

Spoon-Knife History

merge

Frankenstein History

LibreCAD

LibreCAD History

fork

fork

PR accepted
If you don’t believe me

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

and 7 deletions.

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 problem?

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

=> Our algo throws away LibreCAD

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

Legend
- Commit
- Parent → Child

Diagram:
- D: ["repo1", "repo4"]
- B: ["repo5"]
- C: ["repo2", "repo3"]
- E: ["repo6"]
- A: ["repo1"]
Repo group 1

Group 1
- "repo1"
- "repo2"
- "repo3"
- "repo4"
- "repo5"
Repo group 2

D
["repo1", "repo4"]

B
["repo5"]

C
["repo2", "repo3"]

E
["repo6"]

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](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

```python
txt_0 = "hello world my name is babush"
txt_1 = "good morning babush"
```
Building a robust data pipeline

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

- [Apache Airflow](https://airflow.apache.org/)
- [TensorFlow](https://www.tensorflow.org/)
- [Apache Hadoop](https://hadoop.apache.org/)
- [Apache Spark](https://spark.apache.org/)
- [AWS](https://aws.amazon.com/)
Building a robust data pipeline

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

NOPE
Bash

It’s bash all the way

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

![Graph showing the relationship between strings and their popularity](image)

- 
- repos
- 10k
- 100k
- 250k
- GIF89a (1453)
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
Spectral Co-Clustering
Putting everything into production
```latex
scores = database \times query^T
\begin{bmatrix}
  w_{1,1} & \cdots & w_{1,\text{hashes}} \\
  \vdots & \ddots & \vdots \\
  w_{\text{repos,1}} & \cdots & w_{\text{repos,hashes}}
\end{bmatrix}
\begin{bmatrix}
  q_1 \\
  \vdots \\
  q_{\text{hashes}}
\end{bmatrix}
\begin{bmatrix}
  s_1 \\
  \vdots \\
  s_i \\
  \vdots \\
  s_{\text{repos}}
\end{bmatrix}
```

\text{database} \in [0,1]^{\text{repos,hashes}}
\text{query} \in [0,1]^{1,\text{hashes}}
\text{scores} \in [0,1]^{\text{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 \( \text{jaccard_similarity}(A, B) > \text{threshold} \) => delete B
- Complexity \( O(N \times K) \)

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

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.

<table>
<thead>
<tr>
<th>No</th>
<th>Profile</th>
<th>Name</th>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tomonori Shihomi</td>
<td>Remote Code Execution - 1</td>
<td>Cross-Site Scripting (XSS) - 1</td>
</tr>
<tr>
<td>2</td>
<td>Orange Teal(orange_8361)</td>
<td>Insecure Direct Object Reference(DOR) - 1</td>
<td>Cross-Site Scripting (XSS) - 2</td>
</tr>
<tr>
<td></td>
<td><a href="http://blog.orange.tw/">http://blog.orange.tw/</a></td>
<td>Improper Access Control - 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Masato Kinugawa</td>
<td>Cross-Site Scripting (XSS) - 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://twitter.com/kinugawamasato">https://twitter.com/kinugawamasato</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Yuhei Yamauchi</td>
<td>Other - 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://twitter.com/xDV14">https://twitter.com/xDV14</a></td>
<td>Purchase Bypass - 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Authentication Bypass - 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cross-Site Scripting (XSS) - 1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>bagpro(Sergey Toalhin)</td>
<td>Other - 4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Paolo Montesiel (babush)</td>
<td>Other - 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://twitter.com/pumontesiel">https://twitter.com/pumontesiel</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

THANK YOU!
Questions?

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

[YouTube] babushkam
[Twitter] pmontesel