A Vulnerability Introducing Commit Dataset for Java: an Improved SZZ Based Approach

Aladics Tamás, Hegedűs Péter, Ferenc Rudolf
University of Szeged, FrontEndArt Ltd.

ICSOFT, 11-13 July 2022

Introduction

To do meaningful computation in many areas of software development we need **data.**

In the case of **commits** the same things apply! If we aim to:

- Provide meaningful statistics
- Categorize issues
- Train ML models
- etc...

having an abundance of data is mandatory.

Introduction - Vulnerabilities and Commits

Vulnerability-wise commits can be:

- Vulnerability Introducing (VIC): A commit that is responsible for the appearance of a vulnerability

or

- Vulnerability Fixing (VFC): A commit that is part of a fix to a vulnerability

VFC Usage

VFC databases can be used for various purposes:

- Patch presence testing
- Vulnerable code clone detection
- Security patch location

Fortunately, the number of available VFC databases is high.

VFC Databases

- NVD (National Vulnerability Database):
  - uses the CVE reference system
- Projekt-KB
  - maintained by SAP
  - contains Java OSS vulnerabilities referenced by their CVE
  - available through Git

Project-KB structure

a) Repository structure

```
<repo_root>/
  statements/
    CVE-2005-3745/
      statememt.yaml
    CVE-2006-1546/
      statement.yaml
    ...
  LICENSE.txt
  README.md
  ...
```

b) Example statement.yaml file


```yaml
vulnerability_id: CVE-2008-1728
notes:
- text: ConnectionManagerImpl.java in Ignite Realtime
    Openfire 3.4.5 allows remote authenticated users to
    cause a denial of service (daemon outage) by
    triggering large outgoing queues without reading
    messages.
fixes:
- id: DEFAULT_BRANCH
  commits:
  - id: c9cd1e521673ef0cccb8795b78d3cbaefb8a576a
  repository: https://github.com/igniterealtime/Openfire
```

VIC Usage & Databases

VIC databases have various security related uses too, mainly connected to vulnerability prevention:

- Just-in-time vulnerability detection
- Vulnerability localization

However, in contrast to VFC databases, VIC databases are scarce in number.

The goal

Since there are lot of VFC databases it would be beneficial to be able to generate VIC databases from them.

We propose a 2 phase method to do this:

VFC database → VIC candidates → VIC database

SZZ algorithm *Filtering*

Phase 1 - Candidate VIC generation

Briefly, this phase has 3 steps:

1. Parse the VFC database
2. For each commit, run the SZZ algorithm
3. Aggregate the results

SZZ: An algorithm based on the git blame command to detect bug introducing commits. We used an enhanced version of it, called SZZUnleashed.*

* https://github.com/wogscpar/SZZUnleashed

Phase 2 - Filtering

The previous step's output is hardly usable: SZZ generates too many false-positives.
To remedy this, we designed a filtering phase:

1. For each candidate VIC generate a relevance score (shown on next slide)
2. Based on the relevance scores choose the top n candidates
3. Aggregate the results

Relevance score

a) Relevance score

```
relevance_score = 0
for introducing_file in introducing_commit.files:
  fixing_file = fixing_commit.fixing_files.get_by_name(
       introducing_file)

  if fixing_file is None:
    continue

  file_similarity_score = compute_similarity(fixing_file,
       introducing_file)

  contribution_score = fixing_file.base_score *
       file_similarity_score
  relevance_score += contribution_score
```

b) Base score

```
summed_length = sum(fixing_commit.patches)
  for file in fixing_commit.all_files:
    if file.isJava():
      base_score = file.patch.length / summed_length
      fixing_commit.fixing_files.add(file, base_score)
```

Example: CVE-2016-3674

- A vulnerability allowing an attacker to perform an XML external entity attack in multiple components of the XStream project, a Java to XML serializer library.
- This vulnerability occurs in multiple files, such as *Dom4JDriver*, *DomDriver*, *SjsxpDriver*, *StaxDriver*, and 3 more.
- Most of the cases, fixing this vulnerability simply accomplished by setting an appropriate flag, as it can be seen on the next slide.

CVE-2016-3674: Fixing StaxDriver

```
commit sha: c9b121a88664988ccbabd83fa27bfc2a5e0bd139
++- xstream/src/.../io/xml/StaxDriver.java


//Before applying fix
protected XMLInputFactory createInputFactory() {
  return XMLInputFactory.newInstance();
}
```

```
//After applying fix
protected XMLInputFactory createInputFactory() {
  final XMLInputFactory instance = XMLInputFactory.
       newInstance();

  instance.setProperty(XMLInputFactory.
       IS_SUPPORTING_EXTERNAL_ENTITIES, false);
  return instance;
}
```

CVE-2016-3674: After phase 1


```
CVE-2016-3674:
  commitsWithIntroducers:
    c9b121a88664988ccbabd83fa27bfc2a5e0bd139:
      [deec01beaa1bd878f7acda9f035a39238a217ae9,
      bba4bc28e62073f9baac9c58cbc14de958df3b7e,
      72efd4a37f0ab81d2dfeb013d35ec7cbed0510b1,
      ...
      1b0f802b01632954c6ba2a6605592e3e2975f72f,
      4fd39f2f2616d4ea9e1d25d30dc78931be01dfb0,
      c9794d2f905985c8e45fa4d77525c130a5fd0a20]
  repo: https://github.com/x-stream/xstream
```

CVE-2016-3674: Problems with the candidate VICs

- Even though the fix is rather simple, SZZ found 17 candidate VICs
- There is no ranking between them, we don't know how much did they contribute to the vulnerability
- Changes happen in files we are not interested in, for example readme-s or configuration files
- The output is hard to explain

CVE-2016-3674: After phase 2

```
============ CVE-2016-3674 ============

Repo: https://github.com/x-stream/xstream
SHA: c9b121a88664988ccbabd83fa27bfc2a5e0bd139

File base scores:
  SjsxpDriver.java: 0.25982952983227936
  StandardStaxDriver.java: 0.3634863898817707
  StaxDriver.java: 0.2073137200989827
  WstxDriver.java: 0.16937036018696727

Introducing commit SHAs:
  ...
  - 4fd39f2f2616d4ea9e1d25d30dc78931be01dfb0
    - SjsxpDriver.java:
      Similarity: 0.7142857142857143
      Total: 0.18559252130877096
    - StaxDriver.java:
      Similarity: 0.4
      Total: 0.08292548803959308
    - WstxDriver.java:
      Similarity: 0.5714285714285714
      Total: 0.09678306296398129
  Overall score: 0.3653010723123453
```

Results: Two tools to automate the process

1. BugIntroducerMiner: A Java program that iterates over project-kb structured datasets, runs SZZ and aggregates the results
2. FilterBugIntroducer: A well parameterizable Python tool that performs the filtering phase, that also takes care of the many caveats regarding this step:
    - can be instructed to use caching to reduce the huge HTTP traffic
    - easy to modify to different n parameter, similarity function, etc.
    - can generate a log file containing all the relevance, similarity and base scores

Results: New VIC dataset generated from Project-KB

- We chose n = 2 for the top n commits decision
- Contains 564 VFC entries, each of them having at least one VIC corresponding to it (while the unfiltered, SZZ generated database had entries with up to 700)
- 198 open source projects were considered
- More than 110.000 files were checked when calculating the relevance scores

Results: Dataset structure


```
CVE-2008-1728:
  commitsWithIntroducers:
    c9cd1e521673ef0cccb8795b78d3cbaefb8a576a:
    - 6088e21ca06fb62790d9ea02faf8c884302e0cd9
  repo: https://github.com/igniterealtime/Openfire
CVE-2008-6505:
  commitsWithIntroducers:
    04fcefa44bae1263c7cad6986a9dafed67f0164f:
    - e05d71ba329337ba63784555fbbe9bb8e0290543
    - 78e853bcb32ea91b84a070b3d2dc03ab14bc6b23
repo: https://github.com/apache/struts
...
```

Acknowledgement

Thank you for your attention!

This work was supported by the SETIT project (no. 2018-1.2.1-NKP-2018-00004), which has been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the 2018-1.2.1-NKP funding scheme.