Security discussions in open source projects

Supervisor

Dr. Mohammad Ghafari

Student

Noah Bühlmann

February 11, 2020
GitHub

- released in 2008
- extends distributed version control using Git with many other features
- acquired by Microsoft in 2018
- > 100 million code repositories (April 2019)
Issues and pull requests

- **Issues**
  - 'reporting feature'
  - title, description, label(s)
  - can be assigned to user(s) for further investigation

- **Pull requests**
  - 'suggestion feature'
  - title, description, label(s), referenced code
  - someone wants to merge code
Issues and pull requests - lifecycle

ClickOnce fns ql EntryPoint #442

Many ways to evolve:

- Active discussion, change, revisions, agreement, merge/implementation
- Closed due to inactivity
- Closed as duplicate or irrelevant
- Merged without any review
- Remain untouched/open (for years!)
Possible research questions

• How often do people actually report security-related issues?
• How long does it take until a conclusion is reached?
• Who is involved in discussions of security-related issues and to which extent?
Why is it important?

- Open source software introduces opportunities and threats. (Cowan, 2003)
- Availability of the source code helps attackers manipulate software. (Payne, 2002)
- The faster a security issue (vulnerability) is fixed properly, the better.
- Lack of people with experience and proper knowledge may lead to bad implementation.
General approach

Phase 1: Data collection

- Define selection criteria
- Select repositories / projects
- Download data
- Classify issues / pull requests using ML
- Train ML classifier(s) on existing dataset

Phase 2: Data analysis

- Sample relevant subset
- Perform manual qualitative analysis
- Define relevant metrics
- Calculate quantitative metrics
- Qualitative analysis
- Quantitative analysis
Data collection: selection criteria

- Suitable ’nature’ of project
- Actual software development and not personal
- Criteria based on ’The Promises and Perils of Mining GitHub’ (Blincoe et al., 2014)
  - number of commits
  - number of contributors
  - usage of pull requests and issues
  - multiple repositories for one project
  - ...
Data collection: downloading data

- GitHub REST API v3
- GitHub GraphQL API v4
- GHTorrent
Data collection: classification of issues/pull requests

- Preprocessing
  - Text extraction
  - Stemming / Lemmatization
  - Filtering, ranking and security **cross word** analysis (Peters et al., 2019)

- Training machine learning classifiers
  - Five available labeled datasets (Wu et al., 2019)
  - Hyperparameter Optimization (Shu et al., 2019)
  - Multiple classification algorithms were evaluated (Gegick et al., 2010)

- Apply trained classifiers on our data
Data analysis

- Quantitative analysis
  - Define and calculate metrics
  - Other statistical analysis

- Qualitative analysis
  - Sample relevant subset
  - Perform manual review by hand
Possible challenges

• representativeness of selected projects
• class imbalance, cross words and insufficient data
• transfer learning
• required time and knowledge for manual analysis
Thank you!

Questions?
Appendix: FARSEC

FARSEC, a framework composed of a combination of Filtering And Ranking methods to reduce the mislabelling of Security bug reports by text-based prediction models.

*In a Data Matrix each row represents a bug report.*
Appendix: Sources - Part 1

Appendix: Sources - Part 2
