L-017

OSS における脆弱性修正期間の長大化要因の調査
Inspection of Factors Contributing to Prolongation of Vulnerability Fixing Duration in OSS

佐藤 将也
Masaya Sato

1. はじめに

ソースコードが公開され，再配布などがライセンスにより規定されたソフトウェアであるオープンソースソフトウェア（Open Source Software，OSS）が広く利用されている．OSS は，企業によるソフトウェア開発でも用いられており，OSS として公開されたライブラリを含むソフトウェアが広く利用されている．一方，ソフトウェアにおける設計や実装上の不備である脆弱性を利用した攻撃が問題となっている．また，攻撃を目的として OSS に貢献することで攻撃コードを埋め込む事例も観測されている．このように，OSS に限らずソフトウェアにおける脆弱性の有無や攻撃の可能性があると，OSS の活性化を阻害する要因になりうる．このため，OSS の開発者や利用者にとって，当該ソフトウェアの脆弱性の有無を早期に発見できることが望ましい．しかし，OSS の中には脆弱性の発見が遅れたり，脆弱性が報告されたにも関わらず修正まで長期間を有する場合がある．そこで本稿では，OSS における脆弱性修正期間を分析し，脆弱性修正期間の長大化の要因を調査した結果を報告する．

2. オープンソースソフトウェアと脆弱性

OSS とは，ソースコードが公開され，再配布などがライセンスにより規定されたソフトウェアである[1]．OSS の開発では GitHub [2]が利用されることも多い．GitHub は OSS 開発を支援するサービスであり，ソースコードの公開だけでなく，開発者や利用者が当該 OSS に関して議論する機能を持つ．機能追加や変更の要望は Issues という機能を使って議論され，解決されると当該 Issue がクローズされる．Issues では，脆弱性の修正についても議論される．

ソフトウェアにおける脆弱性とは，プログラムの不具合や設計や実装上の不備である．ソフトウェアに脆弱性があると，脆弱性を利用した攻撃によりソフトウェア利用者が被害にあう可能性がある．このため，ソフトウェアには脆弱性が存在しないことが望ましい．しかし，ソフトウェア開発において脆弱性を無くすことは難しい．また，脆弱性が存在しないかどうかを検証することも難しい．そこで，脆弱性を早期に発見し，修正することが求められている．

脆弱性の修正を支援する手法が研究されている[3][4]．文献[3]ではソースコード中の検証不備などを検出して指摘する方法を提案している．これにより，入力値の検証不備などによる脆弱性を防ぐ方法を提案している．文献[4]では，GitHub のコミットを分析しサポートベクタマシンによる分類器を作り，低い誤検知率で脆弱性の疑いのあるコミットを検出する方法を提案している．これらの手法はソースコードを中心に分析しているが，我々はソースコードだけでなく GitHub から取得できるその他の情報も分析に有用であると考えている．

文献[5]では脆弱性を持つパッケージがリリースされるまでの過程を分析し，修正されても即座にリリースされないことや，リリースされても広く行き渡るまで長期間を有することが報告されている．

このように，脆弱性の修正方法や修正期間について多く研究されているものの，修正期間についての分析は少ない．特に，修正期間が長大化する要因については，個別事例の分析が多い．このため，個別のソフトウェアに対して脆弱性修正期間の長大化要因は分析できても，ある脆弱性が発見されたときに，当該脆弱性の修正が長期化し得るか否かを予測することが難しい．そこで本研究では，脆弱性修正期間が長大化する要因について，定量的なデータを用いて推測することを試みる．

3. 脆弱性修正期間の長大化要因の分析

3.1 目的

脆弱性修正期間の長大化要因を分析するために，修正期間とレポジトリ情報および脆弱性情報を用いたクラスタリングを行なう．これにより，修正期間が長いクラスタを探索し，要因となるレポジトリ情報を分析する．

3.2 密度ベースクラスタリングによる分析

脆弱性修正期間（単位：時間）をレポジトリ情報および脆弱性情報と組合せてクラスタリングを行う．レポジトリ情報は，GitHub から取得できる情報として stars, watchers, forks, contributors, open issues, open pull requests, commits, branches, および tags を用いた．また，脆弱性情報として，CVSS v3 base score, exploitability, および impact を用いた．CVSS v3 [8] base score は，脆弱性の利用しやすさ（Exploitability）と影響（Impact）を基に点数が 0 点から 10 点までの範囲で小数点以下第 1 位まで計算される．クラスタリングには DBSCAN を用いた．分析に用いた情報はレポジトリ情報，脆弱性情報，および脆弱性修正期間を取得できた 1,110 件である．脆弱性修正期間は，脆弱性が報告されて修正の完了が観測できた時点までの時間数として計算した．

クラスタリングの結果を図 1 に示す．結果より，いずれの特徴量も脆弱性修正期間との相関は小さいことが分かる．クラスタに着目すると，レポジトリ情報を用いた場合，ほとんどクラスタが作られていないものの，脆弱性修正期間の長い脆弱性においてクラスタが見られる．一方で，脆弱性修正期間の短いものについては，ほとんどが 1 つのクラスタに分類されていることが分かる．また，脆弱性情報を用いた場合，特に Exploitability スコアと Impact スコアでクラスタが構成されていることが分かる．

†岡山県立大学　Okayama Prefectural University

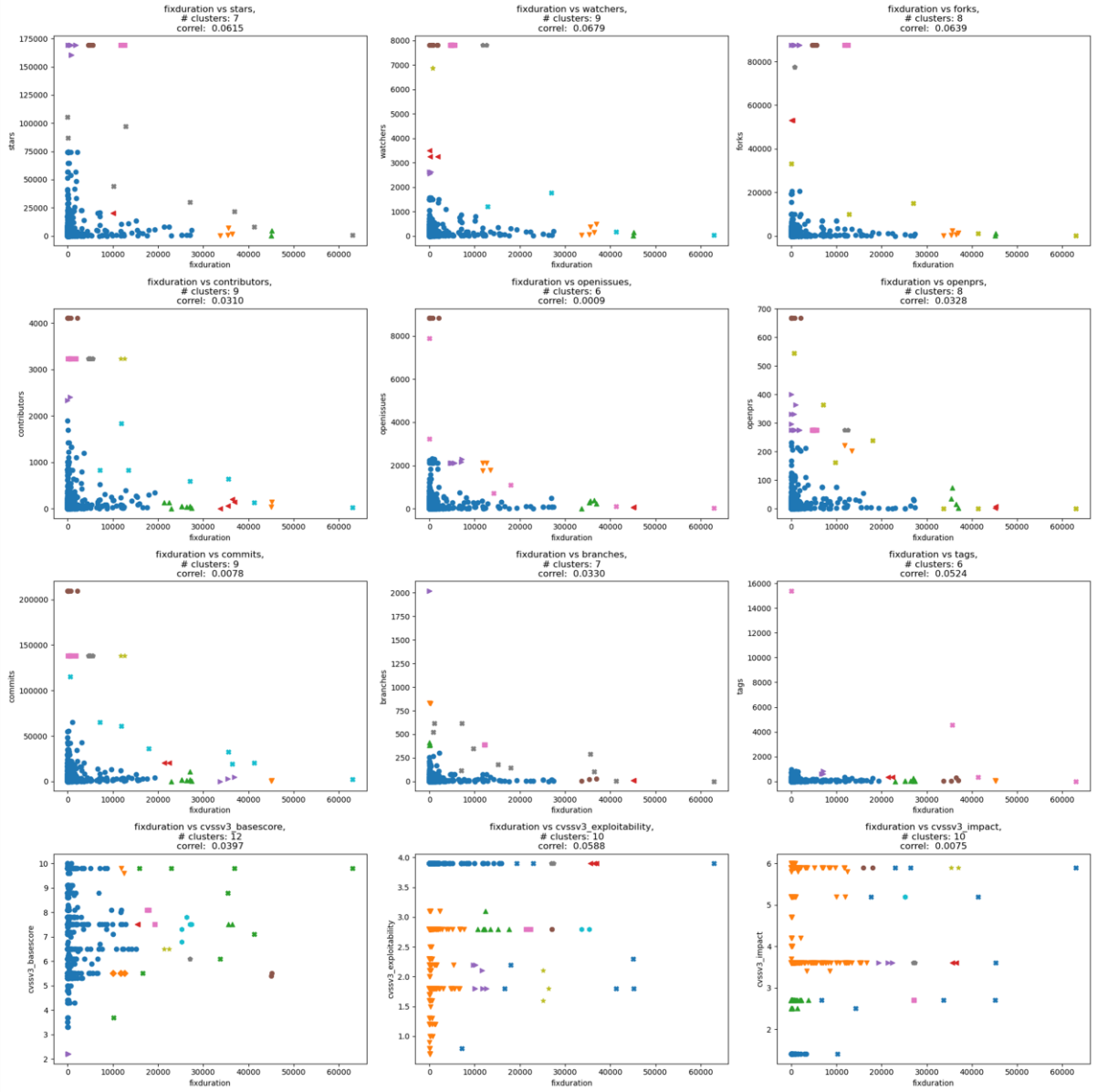

図 1　脆弱性修正期間と stars によるクラスタリングの結果

4. おわりに

オープンソースソフトウェアにおける脆弱性修正期間の長大化要因についての分析結果を述べた．脆弱性修正期間とレポジトリ情報の組合せでは，脆弱性修正期間が長いクラスタが得られた．また，脆弱性情報との組合せにおいてもクラスタが得られた．今後は異なるクラスタリングの適用，および得られたクラスタに注目して分析を進める．

謝辞

本研究を進めるにあたり調査に協力いただいた岡山県立大学情報工学部の齋藤 直弥氏に感謝します．

参考文献

[1] Open Source Initiative: The Open Source Definition (online), available from ⟨https://opensource.org/osd/⟩ (accessed 2024-06-12).
[2] GitHub, Inc. : GitHub (online), available from ⟨https://github.com/⟩ (accessed 2024-06-12).
[3] Yamaguchi, F., Wressnegger, C., Gascon, H. and Rieck, K.: Chucky: Exposing missing checks in source code for vulnerability discovery. Proc. 2013 ACM CCS, pp. 499-510 (2013).
[4] Perl, H., Dechand, S., Smith, M., Arp, D., Yamaguchi, F., Rieck, K., Fahl, S. and Acar, Y.: Vccfinder: Finding potential vulnerabilities in open-source projects to assist code audits. Proc. 22nd ACM CCS, pp. 426-437 (2015).
[5] Chinthanet, B., Kula, R. G., McIntosh, S., Ishio, T., Ihara, A. and Matsumoto, K.: Lags in the release, adoption, and propagation of npm vulnerability fixes, Empirical Software Engineering, Vol. 26, pp. 1–28 (2021).
[6] McInnes, L., John, H., and Steve, A.: hdbscan: Hierarchical density based clustering, J. Open Source Softw, Vol. 2, No. 11, p. 205 (2017).
[7] FIRST: Common Vulnerability Scoring System v3.1: Specification Document , available from ⟨https://www.first.org/cvss/v3.1/specification-document⟩ (accessed 2024-06-12).