6N-06

GitHubにおけるセキュリティバグ報告と修正の調査

中野 大扉† 亀井 靖高† Abram Hindle‡ 佐藤 亮介† 鵜林 尚靖†
†九州大学 ‡University of Alberta

1 はじめに

ソフトウェア開発においてセキュリティバグの修正は重要な作業の一つである．セキュリティバグは情報漏えいやシステム障害に繋がるため，ソフトウェアを開発や保守していく中で継続的に対応しなければいけない．

セキュリティバグを一意に扱うための識別子としてCVE (Common Vulnerabilities and Exposures) が存在する．CVEには各セキュリティバグに対するID，セキュリティバグの内容，公開日などが含まれ，このセキュリティバグについて議論する際に利用される．オープンソースソフトウェアの開発プラットフォームであるGitHub内でも，コミットコメントや問題報告内でCVE-IDは用いられている．

本研究では，CVEの記述を基にGitHub内のプロジェクトからセキュリティバグに関する問題報告を特定し，以下の2つのResearch Questionを基に調査を行う．

RQ1 セキュリティバグの修正方法にはどのような種類が存在するか？

RQ2 セキュリティバグの修正時間はどれくらいであるか？

2 背景

脆弱性が発生してからCVEが発行され，GitHub内で問題報告が行われるまでの概要を図1を基に説明する．脆弱性が発生したプロジェクトは（1）その脆弱性を修正して（2）CVE-IDを発行してもらうよう要求する．要求を基に（3）CVEが登録され，（4）そのCVE情報が脆弱性データベースに格納される．ユーザーは（5）データベースから脆弱性情報を確認し，影響を受けるプロジェクトに（6）問題報告を行う．影響を受けるプロジェクトの開発者は（7）その脆弱性に対応する．本研究では，（3）CVEが登録されてから（6）問題報告が行われ，（7）開発者がそれに対応するまでの時間や対応方法に着目して研究を行う．

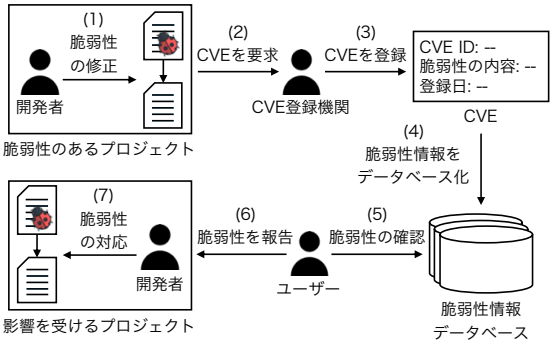

図 1: 脆弱性の発生から問題報告までの概略

関連研究として，Freiらは脆弱性が発生してから修正されるまでの期間や攻撃コードが公開されるまでの期間をCVEを利用して明らかにしている [1]．既存研究では（1）脆弱性の修正情報と（3, 4）CVEに関する情報を基に研究しているのに対して，本研究は（6, 7）CVEに影響を受けるプロジェクトに問題報告が行われ，修正するまでの期間や方法を調べている点で新規性がある．

3 データセット

本研究では，データセットとしてGHTorrentと呼ばれるデータベースを利用する．GHTorrentはGitHubに登録されているOSSリポジトリのデータを集約しているデータベースであり，Gousiosらから提供されている [2]．GitHubにはソフトウェア開発のためではなく，データの公開や保存を目的としたリポジトリが存在する．本研究では，ソフトウェア開発を行っているリポジトリを対象とするためにデータの前処理を行う．データの前処理として，開発人数が10人未満，又はフォークされた数が10未満のリポジトリを対象外とした．フォークとはGitHubから提供されている機能の一つで，リポジトリをコピーする機能のことである．

A Study of Security Bug Reports in GitHub Projects
Daito Nakano†, Yasutaka Kamei†, Abram Hindle‡, Ryosuke Sato†and Naoyasu Ubayashi†
†Kyushu University ‡University of Alberta
{nakano, kamei, sato, ubayashi}@posl.ait.kyushu-u.ac.jp
abram.hindle@ualberta.ca

GHTorrent に含まれている問題報告件数は33,564,779 件であり，そのうち問題報告のタイトル，本文内，または議論内に CVE-ID が含まれている問題報告は 6,463 件であった．このデータから開発人数 10 人以上，かつフォークされた数が 10 以上のプロジェクト内の問題報告を抽出した結果，4,163 件となり，これを本研究の調査対象とした．

4 実験と結果

RQ1 セキュリティバグの修正方法にはどのような種類が存在するか？

研究対象となった問題報告について，どのような方法でセキュリティバグを修正しているかを調査する．セキュリティバグの修正方法にパターンが存在すれば，将来同様の修正を行う際に修正情報を参照できる可能性がある．

調査方法として，研究対象となった問題報告 4,163 件のうち 250 件をランダムにサンプリングを行い，どのような方法で修正をしているかの観点で問題報告を分類した．分類するためのカテゴリを第一著者，第二著者，第三著者の三人による協議により決定した．その後，第一著者が 250 件の問題報告に対して，どのカテゴリに属するかを目視により分類した．

分類のカテゴリは Version Update, Fixing Code, Discussion の 3 つである．Version Update はセキュリティバグのあるライブラリのバージョンを更新することでセキュリティバグを回避しているものであり，88 件存在した．Fixing Code は自身の開発内のコードを変更することでセキュリティバグを回避しているものであり，32 件存在した．Discussion はセキュリティバグについて議論しているものの結果的には修正していないものであり，130 件存在した．

RQ2 セキュリティバグの修正時間はどれくらいであるか？

RQ1 で実際にセキュリティバグを修正している問題報告の修正時間がどれくらいであるかを調査する．開発プロジェクト内で利用しているライブラリにセキュリティバグが発生した際に,「修正には時間がかからないがバグが存在することに気づくのが遅れた」といった場合があれば，バグの存在を通知することでバグの潜在時間を減らすことができると考えられる．本研究では，バグの潜在時間を減らすことができる問題報告が存在するかどうかを明らかにするために，セキュリティバグが公開されてから問題報告が行われるまでの期間（問題報告期間）と問題報告が行われてから修正されるまでの期間（問題修正期間）を調査する．

表 1: 問題報告期間と問題修正期間

期間	カテゴリ	日数				
		最小値	.25	中央値	.75	最大値
報告期間	Version Update	-271	-1	2	39	976
	Fixing Code	-478	-38	-1	69.5	1823
修正期間	Version Update	0	0	0	3	864
	Fixing Code	0	0.5	1	14	610

.25, .75 = 第一四分位，第三四分位

RQ1 の結果から，研究対象の問題報告は CVE-ID が含まれているが，実際にはバグの修正は行われずに議論に止まる問題報告も多いことが分かる．よって，セキュリティバグの修正時間は RQ1 で Version Update と Fixing Code に分類された 120 件の問題報告を対象に調査する．また，問題報告期間は CVE が公開された日から GitHub 内に問題報告された日までとし，問題修正期間は GitHub 内に問題報告された日から問題報告が Closed になった日（解決した日）までとする．

問題報告期間と問題修正期間を表 1に示す．報告期間に 0 日未満の数字があるのは，CVE の情報が一般公開される前からセキュリティメーリングリストのような非公開の方法でバグの修正を行っている場合があるからである．報告期間が 0 以上の問題報告 92 件について問題報告期間と問題修正期間を比較した．その結果，44%（53 件）が問題修正期間よりも問題報告期間の方が長いことがわかった．これらの問題報告は，セキュリティバグの存在を報告することによってバグの潜在時間を減らすことができると考えられる．

5 おわりに

本稿では，GitHub におけるセキュリティバグ修正の種類と修正時間について実験結果を報告した．今後の課題として，同じセキュリティバグを修正しているプロジェクトに着目して，同じセキュリティバグ修正でも問題報告期間や問題修正期間に差が存在するかを明らかにしたい．

参考文献

[1] Frei, S., May, M. and Plattner, U. F. B.: Large-scale vulnerability analysis, *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense, LSAD'06*, pp. 131–138 (2006).

[2] Gousios, G.: The GHTorent dataset and tool suite, *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR 2013*, pp. 233–236 (2013).