On the Use of Dependabot Security Pull Requests

Conference Paper · February 2021

Mahmoud Alfadel
Concordia University Montreal

Diego Elias Costa
Concordia University Montreal

Emad Shihab
Concordia University Montreal

On the Use of Dependabot Security Pull Requests

Mahmoud Alfadel, Diego Elias Costa, Emad Shihab, Mouafak Mkhallalati
Data-driven Analysis of Software (DAS) Lab
Concordia University
Montreal, Canada
{mahmoud.alfadel, diego.costa, emad.shihab, mouafak.mkhallalati}@concordia.ca

***Abstract*—Vulnerable dependencies are a major problem in modern software development. As software projects depend on multiple external dependencies, developers struggle to constantly track and check for corresponding security vulnerabilities that affect their project dependencies. To help mitigate this issue, Dependabot has been created, a bot that issues pull-requests to automatically update vulnerable dependencies. However, little is known about the degree to which developers adopt Dependabot to help them update vulnerable dependencies.**

In this paper, we investigate 2,904 JavaScript open-source GitHub projects that subscribed to Dependabot. Our results show that the vast majority (65.42%) of the created security-related pull-requests are accepted, often merged within a day. Through manual analysis, we identify 7 main reasons for Dependabot security pull-requests being not merged, mostly related to concurrent modifications of the affected dependencies rather than Dependabot failures. Interestingly, only 3.2% of the manually examined pull-requests suffered from build breakages. Finally, we model the time it takes to merge a Dependabot security pull-request using characteristics from projects, the fixed vulnerabilities and issued pull requests. Our model reveals 5 significant features to explain merge times, e.g., projects that have relevant experience with Dependabot security pull-requests are most likely to be associated with rapid merges. Surprisingly, the severity of the dependency vulnerability and the potential risk of breaking changes are not strongly associated with the merge time. To the best of our knowledge, this study is the first to evaluate how developers receive Dependabot's security contributions. Our findings indicate that Dependabot provides an effective platform for increasing awareness of dependency vulnerabilities and help developers mitigate vulnerability threats in JavaScript projects.

***Index Terms*—Dependabot, pull request, dependency, security vulnerability.**

I. Introduction

Modern software systems are increasingly depending on the reuse of code from external dependencies (i.e., packages). While the use of dependencies boosts productivity [1] and software quality [2], it also increases the impact of security vulnerabilities [3]. A security vulnerability in a highly-used dependency may directly impact hundreds of applications, leading to significant financial costs and reputation loss. An infamous example is the Equifax cybersecurity incident in 2017, caused by a web-server vulnerability in the Apache Struts package, which led to illegal access to sensitive information of almost half of the US population (143 million citizens) [4].

The open source community has taken active measures to deal with security vulnerabilities in dependencies. For example, Dependabot is a very popular GitHub bot that creates pull-requests (PRs) to help developers automatically integrate dependency updates and vulnerability fixes into their projects [5]. Dependabot monitors the GitHub Vulnerability Advisories dataset to identify the vulnerable dependencies of the target project. As soon as a dependency vulnerability is identified, Dependabot sends a notification through a PR that updates the vulnerable dependency version to non-vulnerable version that has fixed the security issue, and developers can simply merge the PR to adopt the suggested update. Currently, more than 6 million security and non-security related PRs have been merged in projects from 15 languages supported by Dependabot [6].

Previous work [7] investigated to which extent dependency management tools can convince developers to upgrade out-of-date dependencies, showing that such tools are not yet widely adopted by developers. However, they focus on the general problem of outdated dependencies and do not pay particular attention to security vulnerabilities in dependencies. Given that dependency updates for vulnerability fixes have a critical impact, we specifically focus on studying a very popular dependency tool (e.g., Dependabot) at coping with security vulnerabilities in dependencies. To our best knowledge, little is known about the receptivity and level of adoption of Dependabot security PRs in real open-source software projects.

Therefore, our main goal is to *understand the degree to which developers adopt Dependabot security PRs that tackle dependency vulnerabilities in open source projects.* To achieve our goal, we perform an empirical study involving data from 15,243 Dependabot security PRs that belong to 2,904 active open-source JavaScript projects from GitHub. In the first stage of our study, we examine how often Dependabot security PRs are accepted (merged) and how long it takes to merge them ($\mathbf{RQ}_1$), in order to determine to what extent developers of open-source projects adopt and respond to Dependabot security PRs. We observe that the majority (65.42%) of the Dependabot security PRs in our dataset are merged, often within a day. Still, a significant minority (34.58%) of PRs are not merged.

As such, to understand the motives that led developers to not merge Dependabot security PRs, we qualitatively examine the reasons for Dependabot security PRs being not merged ($\mathbf{RQ}_2$). Our manual analysis identifies 7 main reasons, showing that, by in large, the majority of non-merged PRs are turned-over by Dependabot itself. For example, in 50.8% of the manually studied PRs, Dependabot closes a former security PR in favor of a newer PR that updates to a newer version.

Although the majority of the PRs are merged within a day (RQ_1), we observe a non-negligible proportion of PRs that took longer to be merged. Hence, to understand what would lead open source developers to take a longer time to respond to Dependabot security PRs, we examine the features that influence the time to merge a Dependabot security PR, given that the time is crucial and that the longer a package remains affected, the longer the application that uses it will remain vulnerable to malicious users (**RQ_3**). We observe, using our mixed-effects regression model, five highly important features to explain merge time durations of Dependabot security PRs. While some common wisdom features (e.g., the project activity and the past experience with Dependabot security PRs) are strongly associated with the timespan of the merged PRs, the severity of dependency vulnerability and the level of patch update are not.

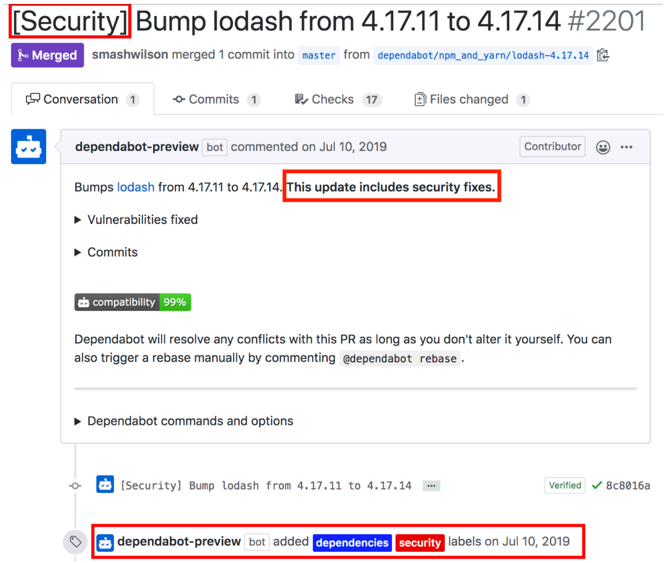
To summarize, this paper makes the following contributions:

- To the best of our knowledge, this is the first work to provide an empirical evidence for understanding developers adoption of Dependabot security automated PRs in open source projects, while also discussing the implications of our findings to practitioners and Dependabot maintainers.
- We qualitatively uncover the possible issues developers could face when adopting Dependabot PRs. Such evaluation can advance the future work, i.e., researchers can direct their efforts to identify the cause of the issues and propose solutions to overcome the limitations.
- We build a logistic regression model that could identify relative importance of various factors explaining merge times of Dependabot security PRs.
- We publish our dataset to help foment further empirical investigations on the related fields [8].

Paper organization. The rest of the paper is organized as follows. Section II describes our study design. Section III presents the results of our study. Section IV discusses how our findings lead to implications to practitioners and future research directions. Section V presents the related work. Section VI presents the threats to validity. Section VII concludes our paper.

II. Case Study Design

Dependabot aims to help developers automatically update their dependencies through PRs. There are numerous reasons to update a dependency, such as making the use of new features, accessing bug fix patches, etc., which led to the creation of millions of Dependabot PRs in open-source projects. Updates that include security issues fixes are among the most critical reasons developers should update their dependencies, as applications frequently depend on packages containing vulnerabilities [9]. Therefore, we focus in our work on studying Dependabot security PRs, i.e., to what extent open source developers adopt Dependabot security PRs to help them keep their dependencies secure. Hence, we first need to identify and collect the dataset of Dependabot security PRs, and use this data to answer the following research questions:



[Security] Bump lodash from 4.17.11 to 4.17.14 #2201

Merged smashwilson merged 1 commit into `master` from `dependabot/npm_and_yarn/lodash-4.17.14`

dependabot-preview bot commented on Jul 10, 2019

Bumps lodash from 4.17.11 to 4.17.14. This update includes security fixes.

▶ Vulnerabilities fixed

▶ Commits

Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also trigger a rebase manually by commenting `@dependabot rebase`.

▶ Dependabot commands and options

`[Security] Bump lodash from 4.17.11 to 4.17.14` ✓ 8c8016a

dependabot-preview bot added dependencies security labels on Jul 10, 2019

Fig. 1: An example of Dependabot security PR [13].

- RQ_1: How often and how fast are Dependabot security pull requests merged?
- RQ_2: What are the reasons for Dependabot security pull requests being not merged?
- RQ_3: What factors are associated with rapid merge times?

Our study examines security PRs created by Dependabot in JavaScript projects. We chose to focus on JavaScript due to its wide popularity amongst the development community [10]. In addition, considering the dynamic nature of JavaScript and the rapidly growing environment (with more than 1.3M packages [11]), the problem of maintaining and updating dependencies is especially challenging, as evidenced by a recent survey of Node.js developers [12]. Hence, dependency management in JavaScript is challenging, which makes Dependabot effectiveness even more crucial.

To perform our study, we leverage the GitHub API to collect security PRs that were created by Dependabot for the purpose of fixing a vulnerable dependency in a JavaScript project.

Obtaining Dependabot security PRs. Dependabot distinguishes minor bug fixes and feature enhancements from security fixes, i.e., whether the dependency update contains a security fix or not. Security PRs submitted by Dependabot contain information related to the vulnerability of the affected dependency, such as the list of vulnerabilities in the security fix. We show an example of a Dependabor security PR in Figure 1. Using the GitHub API, we are able to obtain security PRs by collecting PRs that are: (i) created by Dependabot; (ii) submitted to JavaScript projects in GitHub, and (iii) for the purpose of fixing a security vulnerability (i.e., the PR body refers to a security update). In total, we obtained 36,561 Dependabot security PRs from 6,853 JavaScript projects.

Projects selection. It is known that GitHub contains some toy projects [14], which are not representative of the software projects we aim to investigate. Therefore, once the dataset of Dependabot security PRs is collected, we apply some filtering

TABLE I: Statistics of the 2,904 studied JavaScript projects.

Metric	Min.	Median ($\bar{x}$)	Mean (μ)	Max.
Commits	20	153	465.7	28,486
Age (in days)	146	652	808.3	3,828
Security PRs	1	6	7.3	48

criteria for selecting a set of higher-quality projects. We only include JavaScript projects that are starred, non-forked, and contain more than 20 commits, as recommended by prior studies [7], [14]. After applying these refinement criteria, we end up with 15,243 PRs, which belong to 2,904 open-source JavaScript projects that have at least one vulnerable dependency identified by Dependabot and a security PR was already created for the purpose of fixing it. The affected dependencies contain a set of 167 distinct vulnerable packages. This set contains some popular packages, such as lodash, eslint-utils, jquery, debug, and merge.

Table I shows the descriptive statistics on the selected JavaScript projects in our dataset. Overall, the projects in our dataset have a rich development history and are long-lived projects (median of 153 commits and 652 days of development lifespan), and have received a median of 6 security PRs from Dependabot. Finally, our dataset contains Dependabot security PRs for the period between June 2017 and April 2020. Note that Dependabot launching was on May 26, 2017 [15].

III. Case Study Results

In this section, for each RQ, we present our motivation, describe the approach used, and discuss our findings.

RQ$_1$: How often and how fast are Dependabot security pull requests merged?

In this RQ, we examine the degree to which open source developers are responsive to Dependabot security PRs in the studied projects. Our examination contemplates two main aspects, namely: how many Dependabot security PRs are merged (accepted)? (Section III-A), and how long does it take for these security PRs to be merged? (Section III-B).

A. Acceptance of Dependabot security PRs

Motivation. Given the critical problem of vulnerable dependencies in the current JavaScript landscape, we want to understand how receptive to Dependabot security PRs the open-source projects are. A high adoption rate of Dependabot security PRs indicates that developers value Dependabot contributions and agree with its assessment on the importance of updating their dependencies due to security concerns. Also, given that updating dependencies comes at the risk of breaking the project's own code, the adoption rate shows how often developers are willing to risk breaking their code to use a dependency that is free of vulnerabilities.

Approach. To examine the number of merged PRs, we need first to find the state of each PR in our dataset. PRs have three different states in GitHub: open, merged and closed (i.e., not merged). Open PRs indicate that the PR is not yet processed by developers and the decision about such PRs is not yet taken, hence they are not meaningful for this analysis and have been excluded. To identify whether the PR status is merged (accepted) or not, we extract the value of the key *merged_at* timestamp that is returned from the GitHub API for each PR. For the closed (not merged) PRs, this timestamp is null, while for the merged PRs the merged_at timestamp carries an actual date-time value. After that, we count the frequency of each PR state.

TABLE II: Analysis of the merged and not merged Dependabot security PRs.

Dependabot security PRs	#	%
Total	13,003	100.00%
Merged	8,506	65.42%
Not Merged	4,497	34.58%

Results. The total number of Dependabot security PRs in our dataset after excluding the ones with *open* state is 13,003. **Of the 13,003 examined Dependabot security PRs in our dataset, 65.42% are merged.** Table II shows the proportion of each state of the Dependabot security PRs in our dataset. We observe that the majority of security PRs are merged, indicating that developers are highly receptive to Dependabot security PRs in their projects.

B. Lifecycle of Dependabot security PRs

Motivation. The time needed to process (merge or close) Dependabot security PRs is an important property, as the longer an application remains depending on vulnerable versions of packages, the higher the likelihood of having the vulnerability exploited by attackers. So, to advance our insights, we study whether developers are responsive at merging Dependabot security PRs, i.e., if the time that these security PRs take to be processed is as short as possible. Therefore, we investigate 1) how long does it take to *merge* a security PR since it was first created? and 2) how long does it take to *close* a security PR since it was first created?

Approach. To measure the amount of time it takes for Dependabot security PRs to be processed (merged or closed), we calculate the time difference (in days) between the creation date and the merge date for merged PRs, and the time difference between the creation date and the close date for closed PRs.

Results. Figure 2 presents a violin-plot containing the distribution of the amount of time for the merged and not merged (closed) security PRs, measured in days. From the Figure, we can observe that **the vast majority of the merged Dependabot security PRs are processed within one day (median = 1 day).**

Figure 2 also shows that the closed security PRs tend to take longer time to process than the merged ones, i.e., on median, the closed security PRs took 8 days before being closed. Comparing the merged and closed security PRs using

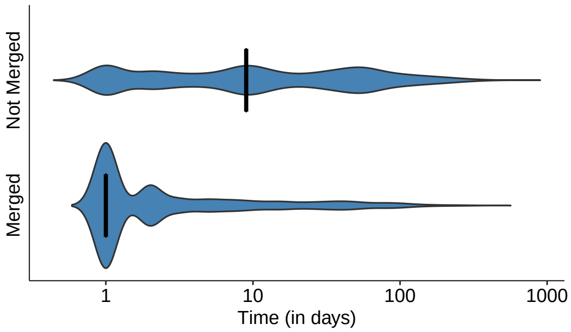
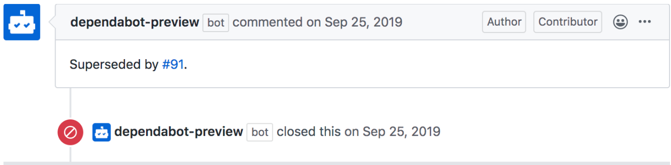


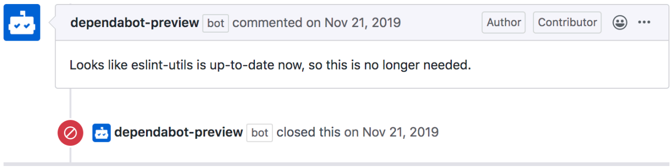
Fig. 2: Violin-plot showing the distribution of the amount of time for Dependabot security PRs to be processed (merged and not merged). Note the logarithmic scale on the x-axis.


dependabot-preview bot commented on Sep 25, 2019 Author Contributor

Superseded by #91.

dependabot-preview bot closed this on Sep 25, 2019

Fig. 3: Example of Dependabot PR closed for being superseded by another Dependabot PR (R1).


dependabot-preview bot commented on Nov 21, 2019 Author Contributor

Looks like eslint-utils is up-to-date now, so this is no longer needed.

dependabot-preview bot closed this on Nov 21, 2019

Fig. 4: Example of Dependabot PR closed because the dependency was already updated (R2).

the unpaired Mann Whitney test [16] shows that this difference is statistically significant (p-value = 2.2e-16), with an effect size (Cliff's: 0.48) for the differences between merged and closed PRs, which is a large size of the effect. This ensures that Dependabot security PRs are either processed and merged fast or left to linger before they are closed without being merged.

> The majority (65.42%) of Dependabot security PRs are merged and integrated in the projects, often within a day. Non-merged Dependabot security PRs take, on median, 8 days to be closed.

RQ$_2$: What are the reasons for Dependabot security pull requests being not merged?

Motivation. While most PRs are merged (as shown in RQ$_1$), a non-negligible share (34.58%) of the PRs are closed (not merged) in the studied projects. It is crucial to understand why such PRs are not merged, to grasp the motives that led developers to dismiss them, especially because such security-related PRs are meant to free open-source projects from known vulnerabilities. In turn, this can be used to motivate improvements at Dependabot, with the aim of increasing its effectiveness. Therefore, we examine why some PRs are not merged, by performing an in-depth manual analysis.

Approach. To find out why Dependabot security PRs are not merged in our dataset, we qualitatively examine them based on the discussion and reviews associated with these PRs. We collect the discussion and review comments related to each closed PR. Out of overall closed PRs (4,497), 1.27% have no discussion or review comments on them, hence, we exclude them from our analysis since it is very hard to judge such PRs without any extra information. The first author manually inspected all remaining closed PRs (4,440) by looking at the discussion and review comments to determine the reason for the closing, and (if possible) summarize the reason for not merging the PR into one sentence. Through this manual analysis, we identified 7 different groups of reasons for the PRs not being merged.

To alleviate the potential bias due to our manual classification for these PRs, we obtain a statistically significant sample of 354 PRs (of the 4,440 PRs) with 95% confidence level and 5% confidence interval. Then, another author independently examined the 354 PRs. Note that the number of comments that span over the discussion of the closed PRs is two, on median, which makes the manual inspection indeed feasible.

To evaluate the agreement between the two authors, we used Cohen's Kappa coefficient [17], which is a well-known statistic that measures the inter-rater agreement level for categorical scales, and takes into consideration the possibility of the agreement occurring by chance. In our categorization of the manually extracted reasons, the level of agreement between the two authors was of +0.96, which is considered to be an excellent agreement [18].

Results. Table III summarizes why Dependabot security PRs are not merged, identified by our manual analysis. Below, we provide more details about each reason.

- **R1. PR is superseded by another newer PR (50.8%):** This is the most common reason for not merging a Dependabot security PR. In this case, the PR is closed because another Dependabot security PR updates the affected dependency to even a newer version that contains fixes to other problems but not necessarily a new vulnerability. In such cases, Dependabot itself closes the former PR in favor of the new and more up-to-date PR. Figure 3 shows an example of R1 [19]. A few cases (1.06%) of superseded PRs are closed by project maintainers where they close a set of PRs and create a single PR that combines all of the changes [20].
- **R2. PR is not merged because the update was applied manually on the dependency file (30.1%)**: Dependabot detects that the fixed version has been applied on the dependency file, hence, it closes the corresponding PR. Figure 4 shows an example of R2 where Dependabot closed the PR that fixes the vulnerable version of the dependency *eslint-utils*. We manually searched for the commit that applied the same fix suggested by Dependabot. In this commit [21], we observe that the same fix version, suggested by Dependabot, was manually added through the commit that also has the

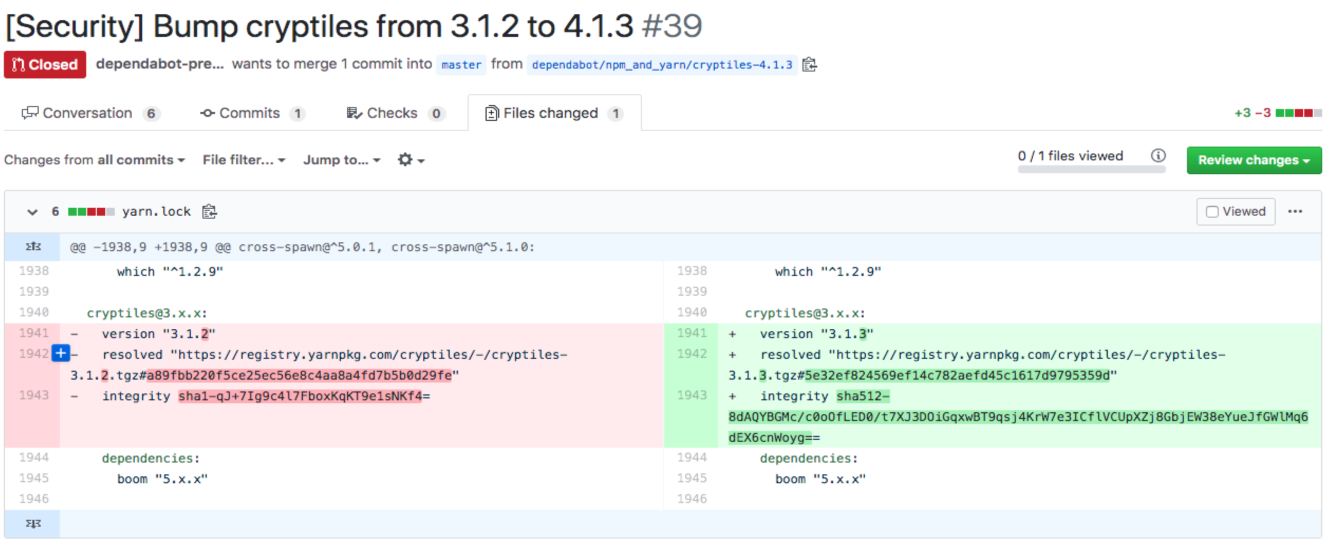
TABLE III: The manually extracted reasons for not merging Dependabot security PRs.

ID	**Reason**	**Description**	**%**	**% Closed by**	
				Dependabot	Others
R1	Superseded	A newer PR contains a newer fix version of the affected dependency	50.8%	49.74%	1.06%
R2	Up to date	The affected dependency is already updated	30.1%	30.1%	-
R3	No longer a dependency	The affected dependency is removed	6.6%	6.6%	-
R4	No longer updatable	The affected dependency has a peer requirement on another dependency	6.4%	6.4%	-
R5	Tests	Tests run failed	3.2%	-	3.2%
R6	Errors	Incorrect implementation for handling the dependency fix in the PR	1.4%	1.4%	-
R7	Quality Requirement	The PR does not comply to the project standards for handling the PRs	1.1%	-	1.1%
R8	Unknown	The PR could not be classified due to lack of information in the discussion	0.4%	-	0.4%

same date as the closed PR date [22].

- **R3. PR is not merged because the affected dependency is removed and no longer exists in the project (6.6%)**: Dependabot will close a PR once the corresponding vulnerable dependency is removed from the project, and hence, the PR is no longer needed [23].
- **R4. PR is not merged due to a peer dependency requirement (6.4%).** Another reason Dependabot closes a PR is when there is a peer requirement between the affected dependency and another dependency. Peer dependencies are a way of specifying dependencies among external packages, when such packages are compatible with specific dependency versions. Hence, to update/fix an affected dependency, its peer dependency should also be updated, which may lead to version conflicts [24]. For example, if the dependency *eslint-config-airbnb@16.1.0* have a peer requirement on *eslint@^4.9.0*, so it is required to update this (eslint) until *eslint-config-airbnb* is updated. In such cases, Dependabot opens a PR to update *eslint-config-airbnb* but later it closes the PR due to having the peer dependency. We found a Github issue in Dependabot repository itself about this problem [25], however, the problem seems not yet properly resolved by Dependabot according to the issue discussion.
- **R5. PR is not merged due to test failures (3.2%).** In such cases, the PR is closed after automated tests have failed during the CI pipeline run [26]. For example, after the Travis tests have failed in this PR [27], the project maintainer closed the PR. When the project maintainer closes the PR, Dependabot will stop notifying the project about the current affected dependency version, however, it opens a new PR when a new fix version of the affected dependency is available.
- **R6. Error in Dependabot (1.4%).** We found cases where the submitted PRs were opened, however, they do not perform the correct fix update, and hence, Dependabot closed such PRs [28]. For example, in Figure 5 we can notice from the PR title that the affected dependency *cryptiles* should be updated from the vulnerable version 3.1.3 to the fixed version 4.1.3. However, Dependabot was not able to resolve the dependency to the fixed version, i.e., the PR commit changes show a different version update than the one should be. This issue is caused by the challenge of resolving dependency conflicts of transitive dependencies. Consider an application that depends on package A, and package A (transitively) depends on package B. Package A has a version constraint for depending on package B (**^1**.0.0) which contains a vulnerability, and the vulnerability was only fixed in another major version (e.g., **2**.0.1) of package B. In this case, Dependabot cannot find a version of package B that complies with the requirement of package A and is not vulnerable. This type of issues render the R6 reason. The has now been fixed by Dependabot maintainers [29].
- **R7. PR does not comply with the project standards for handling PRs (1.1%).** A small share of open-source projects specifies what is called Contributor License Agreement (CLA) that should be signed by the contributor before merging the corresponding PR. In such projects, developers tend to close the Dependabot security PR after it is submitted. To gain more insights about whether such PRs may be still useful for the projects (e.g., project maintainers may manually adopt and apply the dependency fix suggested by the Dependabot PR), we manually analyze a sample of such PRs. In particular, we perform our analysis on 15 PRs of a popular and very active project namely *box/box-ui-elements* [30]. We could find 8 Dependabot PRs that are manually applied to the dependency file by a project maintainer even after closing the Dependabot PRs. For example, in this PR [31], Dependabot suggests updating the vulnerable dependency *atob* from version 2.0.3 to 2.1.2. Although the project maintainer closed the PR, we find that the same dependency update was actually applied as shown in this commit [32], probably to circumvent the licensing issue. One way to overcome the issue of legal side of contributions (i.e., CLA requirements) is to white-list Dependabot in the CLA checker. Some CLA providers (e.g., cla-assistant [33]) allow to white-list specific contributions to a repository.
- **R8. Unknown (0.4%).** In a small minority of cases (0.4%), we could not identify the reason of not merging a Dependabot PR because its discussion and comments provided insufficient information relevant to closing the PR [34].

Overall, the vast majority of the examined PRs (93.9%) are not merged due to four primary reasons related to concurrent modifications of dependencies: **R1 (superseded), R2 (already up-to-date), R3 (no longer a dependency), R4 (no longer updatable)**. Approximately 4% of the PRs are not merged by project maintainers due to factors related to the project process and quality specifications (testing, license agreement). Only 1.4% of the PRs are not merged due to technical errors



[Security] Bump cryptiles from 3.1.2 to 4.1.3 #39

Closed dependabot-pre... wants to merge 1 commit into master from dependabot/npm_and_yarn/cryptiles-4.1.3

```
@@ -1938,9 +1938,9 @@ cross-spawn@^5.0.1, cross-spawn@^5.1.0:
```

```
     which "^1.2.9"

  cryptiles@3.x.x:
-   version "3.1.2"
-   resolved "https://registry.yarnpkg.com/cryptiles/-/cryptiles-3.1.2.tgz#a89fbb220f5ce25ec56e8c4aa8a4fd7b5b0d29fe"
-   integrity sha1-qJ+7Ig9c4l7FboxKqKT9e1sNKf4=
    dependencies:
      boom "5.x.x"
```

```
     which "^1.2.9"

  cryptiles@3.x.x:
+   version "3.1.3"
+   resolved "https://registry.yarnpkg.com/cryptiles/-/cryptiles-3.1.3.tgz#5e32ef824569ef14c782aefd45c1617d9795359d"
+   integrity sha512-8dAQYBGMc/c0oOfLED0/t7XJ3DOiGqxwBT9qsj4KrW7e3ICflVCUpXZj8GbjEW38eYueJfGWlMq6dEX6cnWoyg==
    dependencies:
      boom "5.x.x"
```

Fig. 5: Example of a Dependabot PR closed due to Dependabot's error in the resolved version (R6). As the PR title shows, the affected dependency Cryptiles should be updated from 3.1.2 to 4.1.3, while the diff change shows a different version update from 3.1.2 to 3.1.3.

in Dependabot. Finally, note that the reasons mentioned above are not strictly related to security-related Dependabot PRs.

> The large majority of the closed Dependabot security PRs (93.9%) are turned over by Dependabot due to concurrent modifications of the affected dependencies. Approximately 4.3% of the non-merged PRs are closed by developers due to a specific project's process. Only 1.4% are not merged due to technical issues with Dependabot.

RQ$_3$: What factors are associated with rapid merge times?

Motivation. While most of the merged Dependabot PRs are accepted and integrated within one day (RQ1), there is a sizeable proportion (32.82%) of the merged PRs which took much longer time to be merged. The time taken to handle a Dependabot security PR is crucial given that a quick fix is the only weapon at developers disposal for minimising the risk of the application being affected by external vulnerable dependencies. For example, Heartbleed, a security vulnerability in OpenSSL package, is perhaps the most infamous example. It was introduced in 2012 and remained uncovered until April 2014. After its disclosure, researchers found more than 692 different sources of attacks attempting to exploit the vulnerability in applications that used the OpenSSL package [35], [36]. Hence, in this RQ we aim to study the features that are highly important and associated with the merge time of a Dependabot security PR. Doing so is important to gain understanding of why developers take so long to merge a Dependabot PR that fixes some publicly discovered vulnerabilities.

Approach. Our goal is to study the most important features associated with the merge time of a Dependabot security PR. In particular, we aim to understand the features that are associated with *rapid* merge times. To that aim, we perform a logistics regression analysis that can discriminate whether a Dependabot security PR is merged rapidly or not. Therefore, we first classify merge times into rapid vs. not-rapid. We determine a threshold that discriminates the PRs merge times in our dataset into rapid vs. not-rapid merge times, by evaluating the merge time distribution of the PRs. We find the third quantile (4 days) to be an appropriate threshold. Note that, influenced by prior studies [37], [38], we perform several scenarios for choosing our threshold, i.e., we experimented with different segmentation thresholds (lower quantile, median, upper quantile). For each scenario, we measure the logistics model performance using R-squared (R^2) metric [39]. We use the threshold obtained by the top performing modelling scenario (i.e., the upper quantile). That said, 6,546 PRs belong to the lower 75% of the data points (i.e., those are rapid PR merge times), whereas 1,960 PRs belong to the upper 25% of the data point (i.e., those are not-rapid PR merge times).

To conduct our logistic regression model we first collect a set of features by reviewing the related research on the pull-based software development modelling. Then, we conduct correlation and redundancy analyses to remove highly correlated features because the existence of correlated and redundant features can affect regression models [40]. Finally, we fit a generalized mixed-effects model for logistic regression. These steps are detailed in the following paragraphs.

(i) Features Selection. To determine our set of features, we consult the related literature on the field of pull-based development model, e.g., areas of patch submission and acceptance [41], [42], code reviewing [43], and also dependency vulnerability analysis [44], [45]. The initial list of computed features (described in Table IV) comprises features that span

TABLE IV: The 15 features selected to model the time to merge Dependabot security PRs.

Feature Name	Data Type	Description
PR Features		
changed_lines	Numeric	Number of lines changed (added + deleted) in the dependency file by Dependabot PR
auto_merge	Category	Status of auto-merge method for Dependabot PR. Binary value: True or False
Project Features		
sloc	Numeric	Number of executable source lines of code in the project at Dependabot PR creation time
team_size	Numeric	Number of the active team members in the project at the PR creation time
num_submitted_PRs	Numeric	Number of submitted Dependabot security PRs to the project at the PR creation time
num_accepted_PRs*	Numeric	Number of accepted Dependabot security PRs in the project at the PR creation time
perc_accepted_PRs	Numeric	Percentage of merged Dependabot security PRs in the project at the PR creation time
num_dependencies	Numeric	Number of total proejct dependencies at the PR creation time
num_recent_commits	Numeric	Number of commits in the project during the last month prior to the PR creation time
age (days)	Numeric	Project age at Dependabot PR creation time (i.e., the time interval between project creation time and Dependabot PR creation time)
total_commits*	Numeric	Number of total project commits at the PR creation time
num_issues	Numeric	Number of total project issues at the PR creation time
num_PRs	Numeric	Number of total project PRs at the PR creation time
Vulnerability Patch Features		
severity	Category	Severity of the vulnerability in the affected dependency (Critical, High, Moderate, Low) associated with the Dependabot PR
patch_level	Category	Patch level of the dependency update (Major, Minor, Patch) associated with the Dependabot PR

* Features removed after further step-wise feature selection (e.g., correlation).

over three main dimensions as follows:

PR features. These features attempt to capture the influence of Dependabot security PR characteristics on the merge time. For example, the size of the patch in the PR could affect the merge time [41], [42], i.e., the time needed to examine an external contribution could vary depending on the size of the contribution. Dependabot security PRs have varying size depending on the updated dependencies, such as the number of lines being changed (chanegd_lines) in the dependency file, which may affect the time it takes for developers to review and validate the applied changes. In fact, Dependabot triggers one security PR for each direct vulnerable dependency, by default. However, if the direct vulnerable dependency requires transitive dependencies that are also vulnerable, Dependabot applies additional changes in the same PR, increasing the impact of the changes, e.g., there is more risk in breaking changes when transitive dependencies are vulnerable. Another PR feature is the auto-merge. Dependabot provides an auto-merge feature, which automatically merges Dependabot PRs. A project can enable this feature in case it uses a Continuous Configuration (CI) infrastructure to prevent possible breaking changes. By default, no PRs are auto-merged. Note that we assign the auto_merge as a PR feature, as it can be enabled/disabled during the project lifecycle. Also, enabling the auto-merge feature does not assure that the PR will be merged instantly, given that Dependabot will only merge the PR if the CI tests pass without issues.

Project features. Project features quantify how responsive to Dependabot security PRs the project is. Essentially, such features capture how open the project is to accepting such PRs and its past experience with Dependabot security PRs, by quantifying past Dependabot merged security PRs (e.g., perc_accepted_PRs). Other common wisdom features that can explain the merge time are related to the project size (e.g., sloc, team_size) and maturity of the project (e.g., age). We obtain the project features from previous studies in the field of PR acceptance, as the majority of these features have been successfully used in prior studies [41], [43], [46]–[48] to explain the merge time of a PR.

Vulnerability patch features. The vulnerability patch features quantify the characteristic of the suggested dependency update in the Dependabot PR. There are three main levels of dependency update: 1) patch release indicates backward compatible bug fixes, 2) minor release indicates backwards compatible new features and 3) major release informs developers of backwards incompatible changes in the package release. Therefore, dependency updates that happens at the patch and minor levels are most likely to have minimal impact on the project and can be merged faster by developers. The opposite will happen in updates that bump the dependency to a major release, which have a higher risk of breaking changes and thus may take longer to be merged [44], [45]. Additionally, the severity of the dependency vulnerability is another feature to explain the project response to a security PR [49]. Dependabot builds upon GitHub Advisory dataset [50] to provide a severity level of a dependency vulnerability (i.e., Critical, High, Moderate, and Low).

(ii) Correlation and Redundancy Analysis. The initial list of features included 15 features, shown in Table IV. To make sure that our selected features are not correlated, which could distort their importance in the model, we conduct a pairwise correlation analysis. Specifically, we use the Spearman rank correlation $|\rho|$ metric [51]. A pair of features that have a correlation of $|\rho| \geq 0.7$ should have one of the features removed. We remove 2 features using that cut-off, namely, num_accepted_PRs, total_commits.

Furthermore, we perform RDA (redundancy analysis) on the remaining 13 non-correlated features. A feature can be redundant if it can be modelled using the other independent features. That said, we should eliminate independent features

TABLE V: Results of the mixed-effects logistic model - sorted by χ^2 in descending order.

Feature	Coef.	χ^2	p-value	Sign.$^+$
perc_accepted_PRs	0.63	88.46	< 0.001	***
auto_merge (TRUE)	1.32	64.57	< 0.001	***
num_recent_commits	0.71	32.20	< 0.001	***
num_dependencies	-0.23	7.83	0.005	**
age	0.17	4.32	0.037	*
sloc	-0.09	2.94	0.086	
severity	-	1.49	0.683	
patch_level	-	1.06	0.588	
num_PRs	-0.11	0.55	0.459	
num_submitted_PRs	0.03	0.25	0.617	
changed_lines	0.02	0.17	0.681	
num_issues	-0.05	0.12	0.733	
team_size	0.04	0.09	0.765	

$^+$ Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05

that can be estimated with an $R^2 >= 0.9$ [52]. We observe no redundant features found in the remaining 13 features.

Table IV shows the final 13 selected features (the features without * sign) along with their data type and description. Since the original distributions for most of the features were on different scales, we decided to re-scale the data (standardization scaler) before using them in the model.

(iii) Statistical Analysis. Since our dataset contains PRs from different projects (i.e., PRs merging times vary from one project to the next), we use the generalized mixed-effects logistic model to control the variation between projects. Mixed-effects logistic model, unlike the traditional logistic model, can model the individual differences between the projects by assigning (and estimating) a different intercept value for each project [53], [54]. This allows to capture a project-to-project variability in the dependent feature (PR merge time). We use the *glmer* function in the *lme4* R package to conduct a mixed-effects logistic model.

To evaluate the fitness of our model, we use the R-squared metric for generalized mixed-effects models [39], which describes the proportion of variance considering the *project* variable effect. Also, to measure the explanatory power of the features in the model, and influenced by prior studies [37], [55], we use χ^2 (Chi-Squared). The value of χ^2 indicates whether the model is statistically different from the same model in the absence of a given independent variable according to the degrees of freedom in the model. The higher the χ^2 value, the greater the explanatory power of the feature in distinguishing rapid PR merge times.

Results. Our mixed-effects logistic model achieves a good performance of discriminating the rapid PR merge times of Dependabot security PRs using our determined threshold. The model fits the data well; it explains 67% of the variability in the data (PR merge times) when considering the project variable ($R^2 = 0.67$); and 22% when only considering the independent features without the project variable, showing that the mixed-effect model is more effective at modelling time to merge PRs across different projects.

Table V presents the findings of the features importance derived from the mixed-effects model. All the independent features are ordered on the basis of their explanatory power (χ^2). With each independent feature, we report its estimated coefficient, its explanatory power (χ^2), its p-value, and its statistical significance code (using asterisks) to model the rapid PR merge times. Our results reveal 5 important features to have a strong association with the time to merge a Dependabot security PR. **The top three features are: (1) the percentage of past accepted Dependabot security PRs in the project, (2) the adoption of the auto-merge feature, (3) the level of project activity prior to the PR creation time.** Next, we explain the important features derived from our model.

As shown in Table V, we observe several features that led to merge the Dependabot security PRs fast. For example, the past experience of the project with Dependabot is a major feature that have a strong association with the PR merge time. Projects that have had success in accepting and merging security Dependabot PRs in the past are more likely to merge Dependabot PRs faster in the future, as indicated by the positive coefficient of the `perc_accepted_PRs` in Table V. This also shows that projects that have experienced issues in the past are less inclined to merge the PRs without its due investigation, which may explain the long PR processing time.

Also, enabling the `auto-merge` feature is strongly associated with merging the PRs rapidly. Other highly important features are related to the project activity. For example, the level of project activity, denoted by `num_recent_commits`, has a strong association with rapid merges, i.e., the model indicates that the more active the project, the more likely a Dependabot security PR will be merged within 4 days. Moreover, our model shows that the project age is another important feature that explains the rapid PR merges, although to a lower degree. Projects that have been in development for years are more likely to merge Dependabot PRs within 4 days, as opposed to more recent projects.

Furthermore, we observe that the number of project dependencies (`num_dependencies`) is a feature that correlates with Dependabot security PRs being merged in more than 4 days. This indicates that projects with a high number of dependencies tend to take longer to merge a Dependabot security PR. Projects with many dependencies are more susceptible to dependency vulnerabilities [56], which may lead to an overwhelmingly high number of Dependabot security PRs, taking much longer for developers to address all updates.

Finally, it is also surprising to note that some observed features, such as the vulnerability severity and the dependency patch-level, do not play a significant role in how rapid a Dependabot security PR will be merged. This shows that developers do not necessarily prioritize Dependabot security PRs depending on the severity of the vulnerability or the likelihood of a breaking change (`patch_level`).

The rapid merge time of Dependabot security PRs is directly associated with the project activity level, the project past experience with Dependabot security PRs and the adoption of the auto-merge feature. In contrast, a project with a high number of dependencies is more likely to take longer to process the merges. Surprisingly, neither the severity of the vulnerability nor the risk of breaking changes (patch level) seems to significantly influence the PR merge time.

IV. Implications

In this section, we discuss implications of our findings to practitioners (Section IV-A) and Dependabot (Section IV-B).

A. Implications to practitioners

Open-source projects are highly receptive to Dependabot security PRs. Our results (RQ_1 & RQ_2) show that a large proportion (65.42%) of the Dependabot security PRs are accepted, and 50.8% of the closed (not merged) security PRs are not triggered by developers, but rather by Dependabot itself in favor of more updated dependency versions. The high level of acceptance of Dependabot security PRs indicates that developers are willing to trust external automated tools for important preventive tasks (security dependency updates), given that the tool provides sufficient information for developers to decide. That said, developers should use Dependabot not just to make their dependencies up-to-date but also to keep them secure and vulnerability-free. Dependabot can be seen as a success case to be replicated by tools that assist developers on a variety of tasks like security updates through PRs.

Developers are encouraged to enable the auto-merge feature for improving the merging time of Dependabot security PRs. Our results (RQ_2) show that more than half (50.8%) of the non-merged PRs in our dataset are superseded for not being merged on time. Therefore, we recommend maintainers to review and respond to security updates as quickly as possible to avoid being affected by publicly known vulnerabilities. One way to achieve that is by using the auto-merge feature. Our model (RQ_3) shows the importance of the auto-merge feature. Additionally, our results show that the security dependency updates of Dependabot rarely break the tests in the CI pipeline (3.2%), given the fact that Dependabot issues a PR bumping the current vulnerable version of the dependency to the closest (minimum) non-vulnerable version (to reduce the likelihood of build breakage) [57]. In fact, projects can configure the auto-merge feature to be only enabled for security PRs [58]. That said, developers are better off setting a CI pipeline to automatically merge Dependabot security PRs, particularly in projects that are not in active development or suffer from lack of resources.

B. Implications to Dependabot maintainers

Dependabot needs to properly handle peer dependencies. Our results (RQ_2) show that 6.4% of the closed security PRs are accidentally closed by Dependabot when there is a peer dependency. If a vulnerable dependency A has a peer dependency to B (i.e., the semantic version of the dependency A allows only specific versions to be compatible with the dependency B), creating a PR to update the dependency A would produce version conflicts, effectively leading Dependabot to close the PR after opening it. In such cases, to avoid version conflicts for the peer dependency, the dependency B in the previous example should be updated prior, to be compatible with the new version update of the vulnerable dependency A. At current stage, Dependabot is not fully able to handle such peer dependency updates [25]. Therefore, and given that security updates are essential, Dependabot should find a mechanism to be able to resolve the version conflicts among the peer dependencies in the target project, by updating them to compatible versions.

Dependabot needs to be more efficient for projects with a high number of dependencies. Our model (RQ_3) pinpoints the `num_dependencies` as one of the significant factors for taking a long time to merge a security Dependabot PR. In fact, we have seen several cases (e.g., [59], [60]) where developers have manually consolidated multiple Dependabot PRs into a single PR, to only then update the dependencies all at once. Dependabot can be more efficient by providing ways of grouping PRs to reduce security notification fatigue in large projects. Also, such feature would be more essential in case multiple dependencies need to be updated at the same time or they can break the application.

Dependabot needs to prioritize security updates by more fine-grained analysis of dependency vulnerabilities. Currently, Dependabot provides the maintainers with a way to prioritize receiving notifications for Dependabot security PRs [61]. This is done by using the vulnerability severity level of the suggested security updates. Our model (RQ_3) shows that the security PRs are treated independently of the severity level, indicating a need for a better way of prioritization. A potential improvement to Dependabot is to give priority to updates where the vulnerability part of the dependency is actively used by the project's code. This is admitedly difficult, particularly in dynamically typed languages such as JavaScript, but a conservative approximation can be used to hint developers they need to act fast. Techniques discussed in the literature might be used to achieve this fine-grained prioritization, e.g., SAP organization had recently created a tool that applies static and dynamic analyses to detect and mitigate the use of vulnerable dependencies at the code-level [62], [63].

V. Related Work

The works most related to ours are studies that propose or discuss dependency management tools for security vulnerabilities. Previous studies (e.g., [64], [65]) have shown that projects are slow in terms of responding to security vulnerabilities that are publicly announced, which is sometimes due to factors related to resources and process management. The software development community has proposed several tools that help developers be aware of dependency updates and vulnerabilities. For example, Cadariu [66] developed a Vulnerability Alert Service (VAS), which scans Maven dependencies against vulnerabilities using the Common Vulnerabilities and Exposures (CVE) database. Apiwave [67] is another tool that tracks API migrations in order to help developers be aware of their project dependency updates. One limitation of these tools is that they only send alerts to notify developers about the vulnerable dependencies without being able to automatically fix them or suggest a fix version.

Other works focused on identifying dependency vulnerabilities at a more fine-grain level. For example, Ponta et al. [68] proposed a code-centric tool to detect and mitigate dependency vulnerabilities for Java and Python industry applications. Also, a study by Bodin et al. [63], using extracted methods, show that the code-centric detection tool is viable, although there are challenges related to the JavaScript language and the complexity of the application dependencies. Pashchenko et al. [69], [70] proposed an approach that addresses the over-estimation problem of techniques that report vulnerable dependencies in the Java ecosystem. The authors highlighted that many of the vulnerable dependencies were actually not deployed, and hence, their impact was neglected. Zerouali et al. [71] proposed a tool to analyze vulnerabilities that affect npm dependencies in Docker containers. Di Penta et al. [72] assessed three static analysis tools for detecting source code vulnerabilities in three open-source network systems, and did not find much overlap between the tools' results, suggesting no "silver bullet" vulnerability detection tool.

More specifically, some tools (e.g., [73], [74]) aim to help project maintainers automatically track and update their dependencies. For example, David-DM [73] is a tool that uses what is called "coloured (badges)". The tool checks for outdated dependencies and colours a dependency badge with red, indicating that an outdated dependency version is used. Greenkeeper [74], an automated PRs bot, is another tool that helps developers keep their project dependencies up-to-date by creating PRs that make the required changes for the dependency version update. The work that is most close to ours is the study by Mirhosseini and Parnin [7]. Their work investigated the use of pull requests and badges in the tools David-DM (badges) and Greenkeeper (PRs) to understand whether such tools help developers upgrade outdated dependencies. They analyzed more than 6K GitHub projects that used these tools, and found that projects using the PR tool (i.e., Greenkeeper) tend to upgrade more often than projects that use the Badge tool (David-DM). Nevertheless, the Greenkeeper tool could convince developers of the examined projects to accept only a third of the submitted PRs with a relativity high rate of build breakages (i.e., 25%), indicating the need for better automated dependency tools to convince developers respond to these PRs. The analysis in the study focused on only seven npm packages in the studied projects.

Gousios et al. [41] studied the use of pull-request in the general software development. For example, they found that 80% of typical pull requests are merged within 3 days. Our paper differs from Gousios et al.'s study since we study Dependabot-generated security PRs specially for dependency updates, whereas that study focuses on PRs for software development in general. While we found some overlapping reasons for not merging a PR with the studied from Gousious et al. [41], most of our findings are unique and applicable only to the context of automatic dependency updates.

Our study complements previous works since we specifically focus on studying security updates, i.e., we study a large dataset of Dependabot security pull requests. Moreover, we examine the reasons for Dependabot security PRs being not-merged. Our case study shows that developers make a good use of dependency tools such as Dependabot, responding quickly to the majority of Dependabot security pull requests (less than a day), suffering from a low rate of build breakages. Additionally, our paper adds to the literature through, for example, understanding what factors influence the merge time of a Dependabot security PR.

VI. Threats to Validity

Internal validity: Threats to internal validity concern factors that might affect the casual relationship and experimental bias. In RQ2, we manually analyze the non-merged PRs to identify the reasons of their apparent rejection by developers. This analysis is subjected to the author bias, as every investigator has a subjective method when classifying a PR. We mitigate this threat by asking a second author to independently classify the reasons for not merging and calculate the inter-rater agreement in our methodology (Cohen's Kappa coefficient [17]). The level agreement (+0.96) indicates that our results are more likely to hold.

Another concern is related to the conclusion drawn from the built model by studying the association between the independent and dependent variables. In our work, we study the features that influence the time it takes to merge a Dependabot security PR. To achieve that, we built our model using 13 features that span over three dimensions. However, our set of features are not exhaustive, and other features can be added and show influence for the PR merge times. Still, our model is able to explain 67% of the data variation, which for our purposes is a good initial model for understanding the factors that correlate with the merge time.

External validity: Threats to external validity concern the generalizability of our findings. Our study analyses only JavaScript projects that subscribe to Dependabot. Therefore, our results cannot be generalized to projects of different languages and other ecosystems. Still, given that JavaScript was the first major language supported by Dependabot, it has had a more widespread adoption, which enable us to assess its use on a larger dataset of projects. Furthermore, our methodology can be applied to investigate Dependabot in projects from other programming languages.

VII. Conclusion

This paper conducts an empirical study to investigate the use of Dependabot security pull requests, by examining 15,243 pull requests submitted to 2,904 JavaScript open source GitHub projects. Our results show that a large proportion (65.42%) of Dependabot security PRs are merged, often in one day. Furthermore, our manual analysis leads us to identify that most of the non-merged security PRs (93.9%) are actually closed by Dependabot itself, mostly related to concurrent modifications on the affected dependencies, rather than Dependabot failures. Finally, we build a mixed-effects regression model to understand why some of the pull requests take longer to be merged. Our results reveal 5 important features, e.g., the project past experience with Dependabot security PRs is the most influential feature. We note, however, that the severity of the vulnerability and the risk of breaking changes are not significantly associated with rapid merges. Our findings indicate that Dependabot provides an effective platform to help developers secure their dependencies. Leveraging our findings, we provide a series of implications that is of interest for practitioners and Dependabot maintainers alike.

References

[1] V. R. Basili, L. C. Briand, and W. L. Melo, “How reuse influences productivity in object-oriented systems,” *Communications of the ACM*, vol. 39, no. 10, pp. 104–116, 1996.

[2] W. C. Lim, “Effects of reuse on quality, productivity, and economics,” *IEEE software*, vol. 11, no. 5, pp. 23–30, 1994.

[3] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, “Small world with high risks: A study of security threats in the npm ecosystem,” *USENIX Security Symposium*, 2019.

[4] Equifax, “Equifax releases details on cybersecurity incident, announces personnel changes | equifax,,” accessed on 01/12/2021. [Online]. Available: https://investor.equifax.com/news-and-events/news/2017/09-15-2017-224018832

[5] “dependabot/dependabot-core,” https://github.com/dependabot/dependabot-core, (Accessed on 01/12/2021).

[6] “Dependabot,” https://dependabot.com/, (Accessed on 01/12/2021).

[7] S. Mirhosseini and C. Parnin, “Can automated pull requests encourage software developers to upgrade out-of-date dependencies?” in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 84–94.

[8] Anonymous, “On the use of dependabot security pull requests | zenodo,” https://zenodo.org/record/4437290, (Accessed on 01/13/2021).

[9] "Snyk.io", “The state of open source security 2020 | snyk,” in *Snyk Reports*, 2020, (Accessed on 01/12/2021). [Online]. Available: https://snyk.io/open-source-security/

[10] SOF, “Stack overflow developer survey 2019,” https://insights.stackoverflow.com/survey/2020#overview, 2020, (Accessed on 01/12/2021).

[11] “Libraries.io - the open source discovery service,” https://libraries.io/, (Accessed on 01/12/2021).

[12] R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab, “Why do developers use trivial packages? an empirical case study on npm,” in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017, pp. 385–395.

[13] “[security] bump lodash from 4.17.11 to 4.17.14 by dependabot-preview · pull request #2201 · atom/github,” https://github.com/atom/github/pull/2201, (Accessed on 01/12/2021).

[14] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “The promises and perils of mining github,” in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[15] “Dependabot,” https://dependabot.com/blog/introducing-dependabot/, (Accessed on 01/12/2021).

[16] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The annals of mathematical statistics*, pp. 50–60, 1947.

[17] J. Cohen, “A coefficient of agreement for nominal scales,” *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[18] J. L. Fleiss and J. Cohen, “The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability,” *Educational and psychological measurement*, vol. 33, no. 3, pp. 613–619, 1973.

[19] “[security] bump tar from 4.4.1 to 4.4.13 by dependabot-preview · pull request #91 · slothpixel/ui,” https://github.com/slothpixel/ui/pull/91, (Accessed on 01/12/2021).

[20] “chore(deps): [security] bump axios from 0.18.0 to 0.19.0 by dependabot-preview · pull request #245 · fromdoppler/doppler-webapp,” https://github.com/FromDoppler/doppler-webapp/pull/245, (Accessed on12/30/2020).

[21] “Upgrade dependencies to latest version · steelbrain/babel-cli@c8c9859,” https://github.com/steelbrain/babel-cli/commit/c8c985925c3513f2dd26241a75d71953dd5e1d39#diff-b9cfc7f2cdf78a7f4b91a753d10865a2, (Accessed on 01/12/2021).

[22] “Bump eslint-utils from 1.3.1 to 1.4.3 by dependabot-preview · pull request #85 · steelbrain/babel-cli,” https://github.com/steelbrain/babel-cli/pull/85, (Accessed on 12/30/2020).

[23] “[security] bump stringstream from 0.0.5 to 0.0.6 by dependabot-preview · pull request #33 · 4catalyzer/graphql-validation-complexity,” https://github.com/4Catalyzer/graphql-validation-complexity/pull/33, (Accessed on 12/30/2020).

[24] “[security] bump mixin-deep from 1.3.1 to 1.3.2 by dependabot-preview · pull request #3 · codeparticle/react-visible,” https://github.com/codeparticle/react-visible/pull/3, (Accessed on 12/30/2020).

[25] “No longer updatable · issue #1138 · dependabot/dependabot-core,” https://github.com/dependabot/dependabot-core/issues/1138, (Accessed on 01/12/2021).

[26] “[security] bump eslint-utils from 1.4.0 to 1.4.2 by dependabot-preview · pull request #105 · joshghent/blog,” https://github.com/joshghent/blog/pull/105, (Accessed on 12/30/2020).

[27] “Job #231.2 - joshghent/blog - travis ci,” https://travis-ci.org/github/joshghent/blog/jobs/577015435, (Accessed on 01/12/2021).

[28] “[security] bump cryptiles from 3.1.2 to 4.1.3 by dependabot-preview · pull request #39 · hinaloe/public-toot-viewer,” https://github.com/hinaloe/public-toot-viewer/pull/39, (Accessed on 12/30/2020).

[29] “Js: Handle version resolution for sub-dependencies when not updating dependabot/dependabot-core@b917ac1,” https://github.com/dependabot/dependabot-core/commit/b917ac195748f2d2812071c3cffaf7b77b6b5489, (Accessed on 01/12/2021).

[30] “box/box-ui-elements: React components for box’s design system and pluggable components,” https://github.com/box/box-ui-elements, (Accessed on 01/12/2021).

[31] “build(deps): [security] bump atob from 2.0.3 to 2.1.2 by dependabot-preview · pull request #1521 · box/box-ui-elements,” https://github.com/box/box-ui-elements/pull/1521, (Accessed on 01/12/2021).

[32] “chore: upgrades most dev dependencies (#1753) · box/box-ui-elements@7c8bde4,” https://github.com/box/box-ui-elements/commit/7c8bde43917e9bef50c38ef5e7af3fe168412b1d#diff-8ee2343978836a779dc9f8d6b794c3b2, (Accessed on 01/12/2021).

[33] cla assitant, “cla-assistant/cla-assistant: Contributor license agreement assistant (cla assistant),” https://github.com/cla-assistant/cla-assistant, (Accessed on 01/12/2021).

[34] “Bump gatsby from 2.13.50 to 2.13.52 by dependabot-preview · pull request #860 · prideinlondon/pride-london-web,” https://github.com/PrideInLondon/pride-london-web/pull/860, (Accessed on 01/12/2021.

[35] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey *et al.*, “The matter of heartbleed,” in *Proceedings of the 2014 conference on internet measurement conference*, 2014, pp. 475–488.

[36] “Heartbleed flaw was unknown before disclosure | computerworld,” https://www.computerworld.com/article/2605220/heartbleed-flaw-was-unknown-before-disclosure.html, (Accessed on 01/12/2021).

[37] T. A. Ghaleb, D. A. Da Costa, and Y. Zou, “An empirical study of the long duration of continuous integration builds,” *Empirical Software Engineering*, vol. 24, no. 4, pp. 2102–2139, 2019.

[38] B. Vasilescu, K. Blincoe, Q. Xuan, C. Casalnuovo, D. Damian, P. Devanbu, and V. Filkov, “The sky is not the limit: multitasking across github projects,” in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 994–1005.

[39] S. Nakagawa and H. Schielzeth, "A general and simple method for obtaining r2 from generalized linear mixed-effects models," *Methods in ecology and evolution*, vol. 4, no. 2, pp. 133–142, 2013.

[40] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[41] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 345–355.

[42] P. Weißgerber, D. Neu, and S. Diehl, "Small patches get in!" in *Proceedings of the 2008 international working conference on Mining software repositories*, 2008, pp. 67–76.

[43] P. C. Rigby and C. Bird, "Convergent contemporary software peer review practices," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, 2013, pp. 202–212.

[44] C. Bogart, C. Kästner, and J. Herbsleb, "When it breaks, it breaks: How ecosystem developers reason about the stability of dependencies," in *30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), 2015*. IEEE, 2015, pp. 86–89.

[45] C. Bogart, C. Kästner, J. Herbsleb, and F. Thung, "How to break an api: cost negotiation and community values in three software ecosystems," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 109–120.

[46] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *2015 IEEE/ACM 12th working conference on mining software repositories*. IEEE, 2015, pp. 367–371.

[47] D. M. Soares, M. L. de Lima Júnior, L. Murta, and A. Plastino, "Acceptance factors of pull requests in open-source projects," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, pp. 1541–1546.

[48] M. M. Rahman and C. K. Roy, "An insight into the pull requests of github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 364–367.

[49] A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in *International Conference on Mining Software Repositories*, 2018.

[50] GitHub, "Github advisory database," https://github.com/advisories?page=1, 2017, (Accessed on 01/12/2021).

[51] W. Sarle, "Sas/stat user's guide: The varclus procedure. sas institute," *Inc., Cary, NC, USA*, 1990.

[52] F. E. Harrell Jr, *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[53] B. Winter, "Linear models and linear mixed effects models in r with linguistic applications," *University of California, Merced, Cognitive and Information Sciences*, 2013.

[54] A. J. Lewis, *Mixed effects models and extensions in ecology with R*. Springer, 2009.

[55] S. Ruangwan, P. Thongtanunam, A. Ihara, and K. Matsumoto, "The impact of human factors on the participation decision of reviewers in modern code review," *Empirical Software Engineering*, vol. 24, no. 2, pp. 973–1016, 2019.

[56] A. Gkortzis, D. Feitosa, and D. Spinellis, "Software reuse cuts both ways: An empirical analysis of its relationship with security vulnerabilities," *Journal of Systems and Software*, pp. 1–14, 2020.

[57] "About dependabot security updates - github docs," https://docs.github.com/en/free-pro-team@latest/github/managing-security-vulnerabilities/about-dependabot-security-updates, (Accessed on 01/12/2021).

[58] "Dependabotschedule," https://dependabot.com/docs/config-file/#update_schedule-required, (Accessed on 01/12/2021).

[59] "Update dependencies by andresmoschini · pull request #250 · fromdoppler/doppler-webapp," https://github.com/FromDoppler/doppler-webapp/pull/250, (Accessed on 01/12/2021).

[60] "Dependency updates by edm00se · pull request #49 · edm00se/emoji-transmogrifier," https://github.com/edm00se/emoji-transmogrifier/pull/49/commits, (Accessed on 01/12/2021).

[61] "Configuring notifications - github docs," https://docs.github.com/en/free-pro-team@latest/github/managing-subscriptions-and-notifications-on-github/configuring-notifications#filtering-email-notifications, (Accessed on 01/12/2021).

[62] S. E. Ponta, H. Plate, and A. Sabetta, "Detection, assessment and mitigation of vulnerabilities in open source dependencies," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3175–3215, 2020.

[63] B. Chinthanet, S. E. Ponta, H. Plate, A. Sabetta, R. G. Kula, T. Ishio, and K. Matsumoto, "Code-based vulnerability detection in node. js applications: How far are we?" in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 1199–1203.

[64] R. G. Kula, D. M. German, A. Ouni, T. Ishio, and K. Inoue, "Do developers update their library dependencies?" *Empirical Software Engineering*, vol. 23, no. 1, pp. 384–417, 2018.

[65] A. Zerouali, E. Constantinou, T. Mens, G. Robles, and J. González-Barahona, "An empirical analysis of technical lag in npm package dependencies," in *International Conference on Software Reuse*. Springer, 2018, pp. 95–110.

[66] M. Cadariu, E. Bouwers, J. Visser, and A. van Deursen, "Tracking known security vulnerabilities in proprietary software systems," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2015, pp. 516–519.

[67] A. Hora and M. T. Valente, "apiwave: Keeping track of api popularity and migration," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015, pp. 321–323.

[68] S. E. Ponta, H. Plate, and A. Sabetta, "Beyond metadata: Code-centric and usage-based analysis of known vulnerabilities in open-source software," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2018, pp. 449–460.

[69] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, "Vulnerable open source dependencies: Counting those that matter," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[70] ——, "Vuln4real: A methodology for counting actually vulnerable dependencies," *IEEE Transactions on Software Engineering*, 2020.

[71] A. Zerouali, V. Cosentino, G. Robles, J. M. Gonzalez-Barahona, and T. Mens, "Conpan: a tool to analyze packages in software containers," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 592–596.

[72] M. Di Penta, L. Cerulo, and L. Aversano, "The life and death of statically detected vulnerabilities: An empirical study," *Information and Software Technology*, vol. 51, no. 10, pp. 1469–1484, 2009.

[73] "David, a dependency management tool for node.js projects," http://freestyle-developments.co.uk/blog/?p=457, (Accessed on 01/12/2021).

[74] "Greenkeeper | automate your npm dependency management," https://greenkeeper.io/, (Accessed on 01/12/2021).