Author Response

We thank the reviewers for their insightful comments! We will reorganize the content of the algorithms section as suggested by the reviewers to make the formal treatment clearer. We start by answering the four common questions shared by reviewers and follow with questions made by individual reviewers.

Common questions

Q1. Why is this problem hard? Why not just add proper versioning to each library?

Adding a version tag in the source code of each library is a straightforward solution one can suggest. But this solution can only be accomplished under the leadership of Google, Mozilla, and other companies along with the cooperation of numerous developers to ensure consistency. It has been 20 years since the birth of the web library, but the problem of library versioning has yet to be properly addressed. We urgently need a tool for better versioning. Worse, some libraries have versioning information that is incorrect.

Outside of the web domain, version detection is also a long-standing challenge in both Android analysis and assembly code inference, and has had continuous research in recent years. The algorithm presented in this paper can be applied to any tree-based detection task, and thus can certainly provide insights for versioning disambiguation in other subdomains.

Q2. Why is LDC the only benchmark?

We did not compare PTV against PTdetector because the former is built upon the latter. The direct consequence of this is any library PTdetector can detect PTV can also detect. We will clarify this point in writing and can easily show this empirically as well - though it is an uninteresting result. Library detection is the first step that is accomplished prior to version detection.

According to the PTdetector paper, LDC and Wappalyer are the most popular open-source and commercial tools respectively. LDC can detect versions of around 80 libraries and Wappalyzer around 100 libraries. The tools are designed around the detection of the most commonly used libraries. Although LDC is an old tool, it is constantly updated and as such outperforms Wappalyzer. Additionally, since the publication of the PTdetector paper, Wappalyzer no longer allows for non-commercial use. Therefore, we can only directly compare PTV against LDC in our work.

Q3. Why are there only 64 libraries in our experiment? Can our tool scale to more? Is LDC a strawman?

PTV is automated, so it can detect far more than 64 libs. We restrain our experiment dataset only because we want to fairly compare with LDC to examine the accuracy. In fact, we already published our tool on the Chrome web store with detection ability for **1000+** libs. You can try it out here.

Note that LDC can detect versions over 80 libraries, the discrepancy between that number and the 64 run in the experiment is explained in detail in Sec 5.1.

Some may argue that the number of libraries LDC can detect seems too small. It appears that existing tools like LDC and Wappalyzer focus their detection on the most commonly used libraries in modern websites and expand their detection capabilities to account for deployment trends. We have no formal evidence of this claim and unfortunately, there do not appear to be more robust tools.

Q4. Why didn't our paper show detection runtime overhead?

The runtime detection complexity is not analyzed in Section 3.7, as it is a simple linear complexity, and thus the overhead can be estimated to be not very different from existing tools. Therefore, although we did the detection overhead experiment, it was not put due to space constraints.

As a concrete data point, versioning 1000 libraries yields test results within 500 ms for more than 99% of the websites we tested, with no detectable impact on the site's operation. When versioning the same number (64) of libraries, the overhead difference between PTV and LDC is smaller than the noise in measuring the web application, thus it is not easily measurable. We can add further discussion to the paper to help clarify the baseline performance.

Response to individual reviewers:

Review A

Q5. The approach is limited in the sense that it can only detect versions that it has already seen; it seems tedious to adapt for new versions.

Our tool is fully automated. Therefore, our projected solution is to set a time interval, e.g., one month, and automatically crawl all libraries at this interval to generate the latest detection feature information to ensure the coverage of the latest version.

Q6. Further, it is unclear if the addressed problem is significant as it seems a fix would be to add proper versioning to each library.

See Q1.

Q7. Wonder what the difference is between "minified" and "minimized"

No actual difference. But in the web domain, "minified" is more often used.

Review B

Q8. What is the overall workflow you envision for using PTV in browsers? Do you expect the versions computations to be run each time the page loads or do you think it should be cached and updated frequently?

PTV is a lightweight browser extension. Web users can click the extension button to start the detection when browsing a web page. And our extension can be easily integrated into other automated tools for further analysis. You can try it out here.

Q9. Does the time overhead for version detection affect the browsing capabilities?

See Q4.

Q10. How is the version stubs expected to be stored in the browser and can you determine what libraries need to be versioned?

PTV versions all libraries detected on the web and shows the version result in the extension panel.

Q11. Is CDNJS the largest provider of libraries?

Cdnjs is the only platform that provides web front-end libraries in an integrated way. Previously, each library was served by building its own CDN.

Q12. What do you expect to happens when one uses more than one CDNs?

CDN will not affect our detection result. One library provided by different CDNs is exactly the same.

Q13. I could not follow how to interpret the data presented in Figure 8 and 9.

We will reorganize the data presentation in this part and make it more clear.

Review C

Q14. The paper does not evaluate the soundness of the original technique, PTDetector, versus the new technique, PTV.

PTdetector does not have the version detection ability, so it is out of the scope of the definition of "soundness" in our paper. For more see Q2.

Q15. The paper does not compare the time required to perform library detection with the competing techniques LDC and PTDetector.

See Q4.

Q16. The evaluation also conflates different libraries with their different versions. The evaluation mixes both questions.

The first question is solved in the PTdetector paper, and this paper's work is built on the assumption that libraries are correctly detected.

Q17. This is an average of around 57 versions per library. Is this how the data is distributed?

The version number of each library is close to a normal distribution. We didn't put the experiment due to the article length limitation.

Q18. Can PTV scale to all the libraries in Cndjs?

See Q3.

Review D

Q19. What does sales intelligence mean?

Sales intelligence is the information that salespeople use to make informed decisions in the selling cycle. Similarweb is a sales intelligence example for the web.

Q20. What kind of vulnerabilities were detected?

For instance, once detecting the library version, we can resort to the Synk vulnerability database, which contains known vulnerabilities for specific library versions, and give security advice to web users or developers.

Q21. The experimental comparison is performed with LDC which can recognize versions for only 83 libraries, when the overall domain of libraries is more than > 6K. If the detection rate is so low, isn't LDC a strawman? And if the problem was so compelling, wouldn't there be other tools in this space (given LDC has been in development since 2010)?

See Q3.

Q22 .Compared to widely-used machine-learning-based detection methods"? What are these ML methods? Why not compare with these methods in this paper?

We want to express that ML methods are widely used in detection tasks, but there is no real method proposed for web library detection. Here we will only talk macroscopically about the advantages of our methodological choices - even if ML-related methods became available later, they will not be able to outperform the current method in any respect. We will remove this sentence in our final version.