Author Response

We thank the reviewers for their comments. We start by answering the common questions and then address individual reviewers.

Common questions

Q1. Novelty

Our work is not merely an extension of PTdetector as one of the core contributions of the paper is the novel unique subtree mining algorithm. If, in the future, we find other detection tasks whose features can be abstracted as trees, our method can be applied without any modification.

Q2. Retraining

The overhead of generating features from scratch (retraining) is not high — it takes half an hour for 556 libs as shown in Table.I. Our tool can set a time interval, e.g., one week, and automatically crawl all libraries at this interval to generate the latest detection feature information to ensure the coverage of the latest versions.

Response to individual reviewers:

Review A

Q3. Private Fixes

PTV cannot detect private fixes and we will add this as limitation to the paper. To the best of our knowledge private fixes are not a common practice for web libraries. Existing tools also have a similar limitation. For libraries (and private fixes) using version tags, PTV can enable private fix detection. The PTV distribution that is published on the Chrome web store (not the one used in the experiment), combines the pTree-based method and the tag reading approach. When a version tag exists, we read it directly; otherwise, we apply the pTree method.

Review B

Q4. how to interpret the results of RQ3?

Yes, this setup is artificial and it is designed to more broadly examine PTV and its capabilities. We propose RQ3 to compensate for the limitations of RQ2 in two aspects:

1. In RQ2, we do not have ground-truth; it's hard to know what exact version of the library is loaded. So, we designed RQ3, which has ground-truth, to evaluate the detection correctness of all tools.
2. In RQ2, as shown in Fig.7, the occurrences of different libraries is very uneven in practice, which may result in the most frequently used libraries dominating the experiment results. In RQ3, we test the detectors on each version of each library. According to papers and reports (see [1,2,3] given at the end of the rebuttal), many websites use old versions of libraries. Thus, it is necessary to consider these historical versions.

Q5. More generally, do you see any potential threats affecting the validity of the experimental results and the drawn conclusions?

We have tried our best to provide comprehensive experiments to analyze the performance of PTV. There are three main threats:

1. Libraries that fail to load in the experiments may affect the results.
2. The results of the top-traffic websites may not be able to be generalizable to other websites.
3. Although we believe private fixes (as questioned by reviewer A) in libraries to occur infrequently in practice, we do not know the extent of this practice. PTV cannot detect such libraries.

Review C

Q6. Does PTV frequently return multiple possible versions as the detection results? How many versions can be distinguished by the pTrees?

PTV provides a range of versions as a result. We analyze this with more detail in RQ3; when the PTV's result is consistent with the ground-truth, the range contains only one version. According to table VI, PTV produces a single version as result in 70.8% cases.

Q7. Why is the first subtree not a subtree of T4? Why is not a tree containing path 1-2-4 and 1-2-5 be a unique subtree of T1?

This is a mistake in the figure and description. We will fix this in the final version.

Q8. The ratio of detected_libraries / detectable_libraries achieved by PTV seems lower than existing tools.

PTV can detect many more libraries than were used by the websites in our dataset. Existing tools tend to support only more widely used libraries, thus their ratio is higher.

Rigor4. PTV exhibits much higher overhead, mostly due to the need to collect the path records.

The overhead of PTV is not higher when configured to detect the same number of libraries. In Fig.8, we show the worst case overhead of our tool, where PTV can detect far more libraries (around 5x), while the overhead is only about double. In our published tool, we provide options for users to select how many libraries they want to detect, which allows the overhead to be configurable.

REFERENCE

[1] Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web, Tobias Lauinger et al.

[2] You are what you include: large-scale evaluation of remote javascript inclusions, Nick Nikiforakis, et al.

[3] https://snyk.io/blog/77-percent-of-sites-still-vulnerable/