First, we will address some of the more common concerns by the reviews. One of these was the lack of evaluation against similar tools (particularly the XSS Auditor and NoScript). While we do not report this comparison quantitatively, we mention that there were exploits that our tool blocked and the Auditor did not. Furthermore, we believe a comparison against NoScript would be unfair because its protection model is in direct conflict against our views on maintaining page usability (we do not want to block arbitrary JavaScript, only in carefully specified points).

Another common concern was the amount of manual work required by our solution. While writing signatures certainly requires more effort than automated solutions, we want to emphasize that the majority of the work required to write one is already being done: as mentioned in Section 2, there is an active community which publishes CVEs; this same community could write the signatures without much added effort (just filling out the blanks in our signature language), as they are well aware of the conditions in which the exploits occur. We also want to emphasize that the database entry of the CVE itself is not required for the analyst to write the signature (which helps against zero-day attacks); in particular, the quality of the entry is not paramount to the efficacy of the signature, a concern raised by some of the reviews.

Our system might seem over-reliant on accurate probing/application fingerprinting. As partially discussed in Section 3.1, we acknowledge that this is a hard problem; however, the more popular frameworks often have enough information in the page to identify them, and for hand-crafted sites, the URL suffices.

One last common concern is the lack of empirical data for the difficulty/timing of writing a signature. While we do not report this data, we believe that Section 4 provides a sensible description of the process. We want to point out that this description is in fact more involved than the actual process, as we had to familiarize ourselves with the exploit. This is not strictly part of process as we envision it, since the analysts will already be familiar with the exploit, they will just need to fill the signature fields in.

Review 1 mentions that client-side XSS cannot be detected by a network-level defence. While our implementation does not currently protect against all kinds of client-side XSS, any client-side XSS that takes advantage of, for example, an AJAX request, will be filtered by our extension. Another concern raised by this review is the relevance of the PoC in real-life scenarios. To our filter, the injection is just a dynamic string between two static ones, the contents are irrelevant, with the caveats explained in Sections 2.5 and 2.6.