**Enhancements over the original PLDI conference version**

In our earlier work presented at PLDI 2013, we presented techniques to recover high level functional intermediate representation (IR) from stripped x86 executables. More specifically, we presented techniques to decode the x86 floating point stack, detect complete function prototypes and recover variables and data types in a scalable and accurate manner. In this work, we have extended the original PLDI paper content in the following directions:

1. We describe detailed techniques in recovering the x86 floating point variables in the presence of unresolved indirect branches in the binary. The original paper implicitly assumes that all indirect branches in the binary are resolved correctly and all their targets can be known statically. This part is described in section 3.4.
2. We added formal proofs to all the floating point stack decoding techniques (both the original one presented at PLDI, and the new one presented in this paper). This is shown in section 3.5.
3. We enhanced the overall writing of the floating point recovery part in section 3. This includes separating the problem statement, assumptions, the original technique presented in PLDI, the new technique proposed and the proofs as separate subsections.
4. We added the details of all algorithms used in the function prototypes recovery process. We only had one algorithm in the original PLDI paper and discussed the other algorithms in the text without writing their details. This is presented in section 4 in algorithms 2, 3 and 5.
5. We added a new technique that was not presented before to guarantee the correctness of the external function calls in the rewritten binary. This is presented in the newly added section 5. The new technique includes some modifications to the IR and the introduction of a new “trampoline” function. A discussion about the soundness of the proposed technique is included in the same section 5 (in subsection 5.6).
6. We added a new technique to emit data types correctly in the IR. The technique can emit recursive data types and still ensures correctness of the rewritten IR. This is described in section 6.3. Some practical considerations during the type emission process are added in section 6.4. The soundness and the technique termination guarantees are also added in section 6.5.
7. We modified the results section by adding the percentages of typed IR symbols as a result of our typing techniques at the end of section 7.1
8. We modified the floating point results section (section 7.2) to include the effect of the new proposed algorithms on the IR recovery process as well as adding more detailed (per benchmark) results on the floating point stack recovery process.
9. We added a new section (section 7.4) in the results to show the runtime effect of our new proposed technique that handles external functions on the rewritten binary.
10. The introduction section has a modified contribution list that includes the new contributions described above.

We have also attached the original reviews we received for our main PLDI paper, and our response to them. These can be found in the next pages of this document.