On the Use of Generic Types for Smart Contracts

Fausto Spoto, Sara Migliorini, Mauro Gambini and Andrea Benini

Answers to Reviewer's Comments

Dear Editor,

we would like to submit a new version of our paper "On the Use of Generic Types for Smart Contracts". We thank you and the reviewers for the comments to our submission. In particular, in this revised manuscript, we have carefully considered the reviewer comments and we have provide a detailed answer to their comments in what follows.

We thank you and the reviewers in advance for your work.

The authors:

Fausto Spoto,
Sara Migliorini,
Mauro Gambini
Andrea Benini.

REVIEWER 1

Thank you for your submission. I highly advice you to explain a little bit more about the attack surfaces.

We have expanded the Conclusion section with the following paragraph: “This paper has shown the attack surface due to redefinition of methods with a generic parameter. But another example is the use of instrumented methods to allow access to private state from inner classes: since inner classes are compiled into distinct bytecode classes, the compiler adds non-private accessors to the private state. These accessors cannot be used at source level, but can be called at bytecode level to gain access to private state. This paper does not provide a solution to this other issue, but this further example makes it clear that the attack surface is larger than what described here.”

REVIEWER 3

This manuscript present a real security issue that the authors found while using generics for writing smart contracts that implement shared entities for the Hotmoka blockchain, and they proposed a possible solution based on an appropriate code rewriting.
They are many major comments in here. I bring as much as I can.

We thank the reviewer for the comments. We tried to answer all of them in the following.

1 – The abstract section should be rewritten in order to clearly state the manuscript main focus.

We have rewritten the abstract, stating its focus from its very first line: “This paper shows that generic types (*generics*) are useful for writing more abstract and more general smart contracts, but this comes with some security risks, reporting a concrete security issue found while using generics for writing smart contracts that implement *shared entities* for the Hotmoka blockchain. That issue can be used to steal the remuneration of validator nodes. This paper proposes a patch based on appropriate code rewriting. Namely, smart contracts are pieces of code that are deployed and executed in the context of a blockchain infrastructure in order to automatically enforce some effects when particular events occur. The writing of smart contracts is a complex and critical activity that can benefit from the use of high-level features of programming languages, and generics is one of them. In many programming languages, such as Java, generics are implemented by *erasure*, i.e., replaced by their upper bound type during compilation into bytecode. This is safe at source level, since the compiler takes care of checking that types are correct, before erasure. However, the erased types of the generated bytecode are consequently weaker. In a permissionless blockchain, where every user can call the bytecode of smart contracts installed by other users, these weaker types pose a risk of attack.”

2 – The manuscript should be more organized to make it more interesting for the readers.

We agree that there is always space for improvement and better presentation of the technical material. Still, this comment does not guide us to what exactly should be

organized and made more interesting. A less generic comment would help us. In any case, we have reorganized abstract and introduction following the other comments.

3 – The introduction should be rewritten to clarify its message. Drawbacks of former proposals should be clearly indicated and innovations and new ideas highlighted.

We have reorganized the introduction and placed the following sentence at its beginning, to clarify the message. Note that there are no former proposals, since generics have not been used to write smart contracts up to now, to be best of our knowledge. This is expressed clearly now: “The rules of blockchain transactions are specified by *smart contracts*, that are code written in a variety of programming languages. To the best of our knowledge, none of them allows generic types and, in any case, nothing has been published about the opportunity but also the risks of using generic types for writing smart contracts. The contribution of this paper is exactly to show a real-life use of generic types for an actual smart contract contained in the support library of the Takamaka language, and to demonstrate that a naive use of Java generics can lead to a code security vulnerability that allows an attacker to earn money by exploiting someone else’s work, with both economical and legal side effects. This paper provides a fix to that specific issue, by proposing a re-engineering of the code that forces the compiler to generate defensive checks. More generally, this paper can be useful for the definition of future bytecode languages for smart contract languages, by learning from the weaknesses of Java bytecode, in particular from those related to the compilation of generics.”

4 – Problem statement, objective & contribution are not clear.

The same change to the introduction highlighted at point 3 above clarifies the statemente objective and contribution of the paper now.

5 – If no one has proposed before a solution like the proposed solution, this claim should be highlighted much more. Else, it should be indicated who has done this, and it should be indicated what the innovations of the current paper are.

The same change to the introduction highlighted at point 3 above clarifies that generic types have not been used for writing smart contracts up to now and most programming languages for smart contracts do not even allow the use of generics. We have also added a paragraph at the beginning of the related work section that states that generics have not been used for smart contracts up to now and their related risks have not been studied.

6 – For explaining the related work, the authors need to simply mentions their strategy in generic types or smart contracts, mention the advantages and disadvantages. I did not get the general view of the methods or solutions based the explanation that the authors have provided in this part.

7 – In my view, the literature survey of introduction is quite weak, unfocused and insufficient. The literature review needs to have a flow that leads to the objectives. What is the essential problem of this work? The authors should really explain the drawback of approaches in the related works especially instead of simply stating what they have done. Besides, please explain the main contribution related to previous approaches or solutions.

8 – Section "Related Work" make sure you (i.e., the authors) discuss and cite the most relevant works.

All these comments about the related work section have pushed us to add a paragraph at its beginning, that clarifies that we have really nothing to compare with and we can only put our paper in the context of engineering techniques for strengthening the correctness of software. The new section now starts as follows: "Programming languages specific to smart contracts (such as Solidity) do not have generic types. General-purpose programming languages do have generic types in most cases, but are much less frequently used for writing smart contracts. In any case, we are not aware of any scientific work on the use of generic types for writing more generic smart contracts, nor of any study on the security risks, and their solutions, that this implies for the resulting smart contracts. From this point of view, the present paper has no direct literature to compare with. It is possible instead to insert this paper in the broader context of software correctness and security. Namely...".

Moreover, we have tried to contextualize the proposed solution with respect to the described technique, asserting that it could be used in the future to automatically overcome the vulnerability or guiding the definition of the next generation of programming languages for smart contracts.

9 – Avoid using single person pronouns like "we" throughout the manuscript.

We have chased and removed as many as possible of them, in the whole paper. Currently, only three "we" remain in the manuscript, in Section 8 "Related Work", since they state our personal view.

10 – Some terms are just used without explaining the meaning. It is not enough to bring a concept and just bring a reference. The authors need to explain a little bit about it.

We have tried to imagine which are such terms. The manuscript is expected to be read by somebody working in the blockchain community, hence we do not explain all terms in that area. Yet, if there are specific terms to explain, we are happy to do it but this comment does not help us identifying the exact explanations that are missing.

REVIEWER 4

The manuscript is well written. I accept the paper in its current form.

We would like to thank the reviewer for this comment.