Tainting Checker Overhaul

GSoC Project Report

- ## Overview

Checker Framework is an intricate type-checking verification tool. The Checker Framework enhances Java's type system to make it more powerful and useful which lets software developers detect and prevent errors in their Java programs at compile time. It is based on the idea of pluggable type-checking which involves replacing a programming language's built-in type system with a more powerful, expressive one.

My project, as a student developer at Google Summer Of Code 2020, was to improve the Tainting Checker in terms of functionality. The Tainting Checker annotations have minimal built-in semantics and are too generalized to be used for any specialized data types (like SQL, HTML or OS statements).

The aim of my project was to overhaul the tainting checker; to make the annotations less ubiquitous and easily customizable by the user.

- ## Project Work Summary

The main objective of my project was to increase the Tainting Checker functionality and customizability. Over the last 3 months, I have made several commits to make the Tainting Checker annotations less generalized, e.g. adding a string array argument to the tainting annotations. Apart from this, I also added several additional functionalities to the Tainting Checker, like indirect information flow checks and addition of a '-Aquals' option in Tainting Checker for custom hierarchies.

Apart from the Tainting Checker, I have also made several contributions to the Aliasing Checker. This includes checks for unique parent class constructors and

support for @Unique classes, all of which act as a fix to issue#3313 (https://github.com/typetools/checker-framework/issues/3313). One significant contribution was the addition of the @Linear annotation in the Aliasing Checker which is an idea inspired from the deleted Linear Checker in the Checker Framework. This extension was done to combat issues regarding the java.util.Stream where one could use a reference to a Stream twice without any warning/error raised from the compiler.

I also fixed several issues in the Checker Framework, namely the issue#1395 (https://github.com/typetools/checker-framework/issues/1395) and issue#3033 (https://github.com/typetools/checker-framework/issues/3033), regarding the dump on error issue and the instance of @Nullable class issue respectively.

For more details on these issues, please see Pull requests, commits and work done

The last 3 months have been the most challenging (and, at the same time, fun) period in my early software engineering career. The Checker Framework is by far the largest open source project I have worked on and none of my contributions would be possible without the help of my mentors and my fellow participants.

- ## Pull requests, commits and work done

(I) typetools/checker-framework

Repository Link : https://github.com/typetools/checker-framework

1. Tainting Checker Overhaul : An overhaul of the Tainting Checker to make the annotations more specialized and customizable. Includes addition of string array arguments to the annotations, a '-AindirectInfoFlow' flag to enable indirect information flow checks and, '-Aquals' and '-AqualDirs' subtyping options to facilitate custom tainting annotations and hierarchies.

PR Link : https://github.com/aditya3434/checker-framework/pull/1

PR Status : Open

2. Linear Checker : An extension of the Aliasing Checker which includes the addition of the @Linear annotation. @Linear objects are similar to @Unique objects except that their reference can only be used in a method invocation once, after which they become unusable. This extension was done as a result of the java.util.Stream case study.

PR Link : https://github.com/aditya3434/checker-framework/pull/3

PR Status : Open

3. Aliasing test constructor declarations : An addition of a test case to check @Unique constructor declarations of the Object, String, StringBuffer and Exception classes in the annotated jdk. Part of a fix for issue#3313

Issue Link : https://github.com/typetools/checker-framework/issues/3313

PR Link : https://github.com/typetools/checker-framework/pull/3337

PR Status : Merged

4. Aliasing explicit annotation fix : A fix for issue#3313. Makes the Aliasing Checker check for @Unique object classes instead of just checking for explicit @Unique annotations, when deciding whether a reference can be leaked.

Issue Link : https://github.com/typetools/checker-framework/issues/3313

PR Link : https://github.com/typetools/checker-framework/pull/3336

PR Status : Merged

5. @Unique object return issue : An issue raised by me after the explicit annotation fix (PR#3336). @Unique objects, when returned, don't raise an error if the return type of the method doesn't match, when they should ideally raise a "unique.leaked" error.

Issue Link : https://github.com/typetools/checker-framework/issues/3486

Issue Status : Open

6. Dump on error fix : A fix for issue#1395. Addition of a '-AdumpOnErrors' flag which, when enabled, stores the stack trace at the instance where the error in the program occurred. This was done to fix the issue where the stack trace printed by the '-doe' flag was incorrect for compound checkers.

Issue Link : https://github.com/typetools/checker-framework/issues/1395

PR Link : https://github.com/typetools/checker-framework/pull/3406

PR Status : Merged

7. Validate instanceof types : A fix to issue#3033, regarding prohibited use of @nullable annotation on the type of an instanceof. The added code will issue an error if the type of the instanceof is explicitly annotated as @nullable and will issue a warning if it is explicitly annotated as @nonnull. The fix also works for other checkers and also refines the expression type.

Issue Link : https://github.com/typetools/checker-framework/issues/3033

PR Link : https://github.com/typetools/checker-framework/pull/3481

PR Status : Open

(II) typetools/jdk

Repository Link : https://github.com/typetools/jdk

1. Annotating Object class in jdk : Annotating Object class as @Unique in the jdk as part of a fix to issue#3313.

Issue Link : https://github.com/typetools/checker-framework/issues/3313

PR Link : https://github.com/typetools/jdk/pull/52

PR Status : Merged

2. Annotating stubfile classes in jdk : Annotating stubfile classes like Object, String, StringBuffer and Exception as @Unique in the jdk in order to remove the astub file in the Aliasing Checker.

PR Link : https://github.com/typetools/jdk/pull/57

PR Status : Merged

(III) codespecs/daikon

Repository Link : https://github.com/codespecs/daikon

1. Suppressing Warnings for daikon test : Adding a @SuppressWarnings line in OneOf.java.jpp to just check the data type (and not the annotations) of the state variable in the boolean state_match() method. This was done to bypass the instanceof error and pass the daikon tests in the pull request #3481 in typetools/checker-framework (typetools/checker-framework#3481).

PR Link : https://github.com/codespecs/daikon/pull/272

PR Status : Closed

- ## Documents and Case studies

While working on different aspects of the Checker Framework, I performed various case studies, documented ideas and also kept a weekly progress log. Some of these were ideas that were either fully or partially implemented, or scrapped entirely, although all of them provide some interesting perspective and were discussed by me and my mentor thoroughly. The links to these documents are given below:

GSoC Weekly Progress Log :
https://docs.google.com/document/d/14NuO4M1hprx-24-icaib3Sth3CE6pqocxVZiCla6dn8/edit?usp=sharing

Phase 1 Monthly Report :
https://docs.google.com/document/d/1tgTdAInjGjxcZNL8vVQ4wwnw8CKAUTTSemVdaulG10M/edit?usp=sharing

Phase 2 Monthly Report :
https://docs.google.com/document/d/1sXNPcpfMtY_6nC5wa4fL4QwmRMCQ-PTDKZ1f3awvREA/edit?usp=sharing

java.util.Stream issue reference :
https://www.baeldung.com/java-stream-operated-upon-or-closed-exception

Tainting Checker String Array Arguments and Hierarchy :
https://drive.google.com/file/d/12HScfLI5PXpJt4GLF_Zg1fE1CDL2PGug/view?usp=sharing

Tainting Checker Subtype Idea :

https://docs.google.com/document/d/1zEiyP89OuuAdx0kuVhdqVCoPUJn9WJuwRIJHLTItoM0/edit?usp=sharing

Tainting String Argument and Security ideas : https://docs.google.com/presentation/d/17x5vtK0w9Cd-xVgMGnJAsh7ZdjpBMMT9WXXSZkorrLU/edit?usp=sharing

- TODO

There are still few additional features that can be implemented along with the ones mentioned above.

1. At the moment, the Checker Framework only verifies the condition to check for indirect information flow and not the 'then' and 'else' blocks of the condition. To further validate these checks, the Checker Framework needs to take these blocks into consideration before raising the error. One approach is to taint the whole block and then check whether a tainted object is being used in a high level function.

2. My explicit annotation fix also revealed another issue with the Aliasing Checker which needs attention. @Unique objects, when returned, don't raise an error if the return type of the method doesn't match, when it should ideally raise a "unique.leaked" error. This led me to open Issue#3486 (@Unique object return issue) in typetools/checker-framework, which is currently not fixed.

Issue Link : https://github.com/typetools/checker-framework/issues/3486