Security in Software Applications 2024/2025
Third Project

The project is divided into two parts. You need to:

- Complete at least the first part of the assignment.
- Provide a report in `pdf` describing:
  - In a few rows how the tool you are using works, i.e. describe what it means to do fuzzing, and how Echidna enforces the properties required by the developer.
  - The meaning of the code you have written to check the properties, i.e. describe the meaning of the code used to describe your properties.
  - Include screenshots of the output of the tool before and after you found and corrected the issue.

  The report should not be too long. No more than 10 pages (not restrictive).

  NOTE: I should be able to fully understand the meaning of your code without inspecting the code or installing and running the tool you used. A missing report automatically leads to assignment failure.

You must use Echidna as a property-based fuzzing tool: `https://github.com/crytic/echidna`.

I further recommend a pass of the code into Remix ide (`https://remix.ethereum.org/`) to fix any error or warning triggered by the Remix static analyzer.

Points below will be summed to the final grade.

Part 1 (max 3 points)

In the file Person.sol the contract is used in an information system at the welfare office to record information about individuals. The welfare office requires the following property, easy to overlook: if person x is married to person y, then person y should of course also be married and to person x.

Your assignment is to:

1. add *invariants* to the code to express the properties above using one of the proposed tools, plus any other sensible properties you can think of;
2. use the chosen tool to detect possible violations;
3. Fix the complaints by
   (a) correcting the code, or
   (b) adding a (sensible) precondition for the method, using require(”condition”), or
   (c) adding an invariant for another property that can help in the verification.

Hints

- It is easiest to introduce one invariant at a time, then fix the errors detected and only then move on to the next one.
- If the tool produces some warnings, i.e., if the tool thinks that some property is not satisfied by the code, this can have several causes:
  - There is an error in the code. For example, an assignment to some field may be missing.
  - There is an error in the specification. For example, maybe you have considered an "and" in a specification where you meant "or", maybe you have written age > 18 when you meant age ≥ 18 in some constraints.
  - There may be some properties missing but needed by the tool for the verification. Note that the tool does not know any of the things that may be obvious to you – for example, that if some person x is married to y then y is also married to x – and such properties may be needed for verification.

To stop the tool from complaining, you can:

- correct the contract code; for the exercise here, this should only involve adding some simple assignments to the offending method;
- Correct the specification.

Part 2 (max 2 point)

The welfare system has rules to modify allowances based on marriage and age:

1. Every person receives a default subsidy of 500 until age 65.
2. After the age of 65, the default subsidy increases to 600 if unmarried
3. Married persons receive each the default subsidy reduced by 30%

Add invariants that express these constraints, and, if necessary, fix/improve the code to ensure that they are not violated.